UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE PATENT TRIAL AND APPEAL BOARD

In re
Inter Partes Review of
U.S. Patent No.: 7,490,151
Inventors: Munger *et al.*
Issue Date: Feb. 10, 2009

Trial Number: IPR2013-00376

Title: Establishment of a Secure Communication Link Based on a Domain Name Service (DNS) Request

Attorney Docket No. 3959/5002

Attn: Mail Stop PATENT BOARD
Patent Trial and Appeal Board
United States Patent and Trademark Office
PO Box 1450
Alexandria, Virginia 22313–1450

Declaration of Russell Housley Regarding U.S. Patent No. 7,490,151

I, Russell Housley, do hereby declare and state, that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Dated: 23 June 2013

Russell Housley

I, Russell Housley, declare as follows:

I. <u>INTRODUCTION</u>

A. Engagement

1. I have been retained by counsel for New Bay Capital, LLC as an expert witness in the above-captioned proceeding. I submit this declaration in support of the Petition for Inter Partes Review (hereinafter "the Petition") of claims 1 and 13 of United States Patent No. 7,490,151 (hereinafter "the '151 Patent" – Exhibit 1001), filed in the United States Patent and Trademark Office on behalf of New Bay Capital, LLC.

B. Background and Qualifications

2. I am the founder and owner of Vigil Security, LLC, which I founded in 2002 to help customers design and implement diligently watchful security solutions. I provide consulting on security protocols, security architectures, and Public Key Infrastructure (PKI). Over the last ten years, I have performed security and vulnerability analyses of various communications architectures and security policies based on known threats and proposed certification criteria.

3. Since March 2013, I have served as the chair of the Internet Activities Board (IAB), which is a voting member of the IAB as well as a non-voting

member of the Internet Engineering Steering Group (IESG), a voting member of the IETF Administrative Oversight Committee (IAOC), and a Trustee for the IETF Trust. Since May 2013, I have served as a member of the Internet Research Steering Group (IRSG).

4. From March 2007 to March 2013, I served as the chair of the Internet Engineering Task Force (IETF). I managed the open and transparent technical standards process for the Internet.

5. From March 2003 to March 2007, I served as the IETF Security Area Director, making me a member of the IESG. As such, I provided leadership to many working groups that were developing security standards for the Internet, including the Public Key Infrastructure using X.509 (PKIX), IP Security (IPsec), Transport Layer Security (TLS), Secure MIME (S/MIME), Domain Keys Identified Mail (DKIM), Long-Term Archive and Notary Services (LTANS), and Multicast Security (MSEC) working groups.

6. Prior to accepting the Area Director position, I chaired the IETF Secure MIME (S/MIME) Working Group, and I contributed to several cornerstone Internet PKI standards (including RFC 5280). In November 2004, I was recognized by the IEEE 802.11 working group for my contributions to IEEE 802.11i-2004, which fixes the severe security shortcoming of the Wired Equivalent Privacy (WEP). I provided major contributions to several security protocols, including the

Cryptographic Message Syntax (CMS), SDNS Security Protocol 4 (SP4), SDNS Message Security Protocol (MSP), IEEE 802.10b Secure Data Exchange (SDE) Protocol, and IEEE 802.10c Key Management Protocol.

7. I have worked in the computer and network security field since 1982. Before starting Vigil Security, I worked at the Air Force Data Services Center (AFDSC), Xerox Special Information Systems (XSIS), SPYRUS, and RSA Laboratories. My security research and standards interests include security protocols, certificate management, cryptographic key distribution, and high assurance design and development practices. I have been active in many security standards organizations, and my recent focus has been on the Internet Engineering Task Force (IETF).

8. I have served as the Chair of CertiPath Policy Management Authority, where I assisted with the transition from SHA-1 to SHA-256. I also provided technical and policy advice to the WiMAX Forum Policy Authority for the PKI that is used to authenticate WiMAX Devices and the separate PKI that is used to authenticate the AAA servers within a WiMAX network.

9. I am a Consultant to the U.S. Government. I helped with Crypto Modernization activities, especially in the areas of secure firmware loading, trust anchor management, public key infrastructure, and key management infrastructure.

10. I am a member of the Advisory Board for the Georgetown Center for Secure Communications (GCSC) at Georgetown University, the Security and Software Engineering Research Center (S2ERC) at Georgetown University, and the Center for Information Assurance at the University of Dallas, Graduate School of Management. I am a technical advisor to Penango.

11. I received a Bachelor of Science in computer science from Virginia Tech in 1982, and I received a Master of Science degree in computer science from George Mason University in 1992.

12. I am the co-author of two books: *Implementing Email and Security Tokens: Current Standards, Tools, and Practices*, published by John Wiley & Sons in 2008, and *Planning for PKI – Best Practices Guide for Deploying Public Key Infrastructure*, published by John Wiley & Sons in 2001.

13. I am the inventor of five U.S. Patents:

- US Patent 6,003,135: Modular security device
- US Patent 6,088,802: Peripheral device with integrated security functionality
- US Patent 6,904,523: Method and system for enforcing access to a computing resource using a licensing attribute certificate
- US Patent 6,981,149: Secure, easy and/or irreversible customization of cryptographic device
- US Patent 7,356,692: Method and system for enforcing access to a computing resource using a licensing attribute certificate.

14. A copy of my curriculum vitae, which describes in further detail my qualifications, responsibilities, employment history, and publications is attached to this declaration as Appendix A.

C. Compensation and Prior Testimony

15. I am being compensated at my normal consulting rate for my work and testimony in this matter. I also am being reimbursed for reasonable and customary expenses associated with my work and testimony in this matter. My compensation is not contingent on the outcome of this matter or the specifics of my testimony and in no way affects the substance of my statements in this Declaration.

16. I have no financial interest in Petitioner or in the '151 Patent.

17. I have never testified in Federal District Court, but I testified in the U.S. International Trade Commission on January 13, 2012. Also, I was deposed on May 31, 2005 for a civil action in the U.S. District Court for the Eastern District of Virginia, Alexandria Division.

D. Information Considered and Right to Supplement

18. My opinions are based on my years of education, research and experience, as well as my investigation and study of relevant materials. In forming

my opinions, I have reviewed and understand the materials referred to herein or listed in Appendix B.

E. Availability for Cross-Examination

19. In signing this Declaration, I recognize that the Declaration will be filed as evidence in a contested case before the Patent Trial and Appeal Board of the United States Patent and Trademark Office. I also recognize that I may be subject to cross-examination in the case and that cross-examination will take place within the United States. If cross-examination is required of me, I will appear for cross-examination within the United States during the time allotted for cross-examination.

II. <u>LEGAL STANDARDS AND ANALYSIS</u>

20. I have reviewed and understand the specification and claims of the ‘151 Patent.

21. I believe a person of ordinary skill in the art in the field of the ‘151 Patent would be someone who, prior to February 2000, was familiar with TCP/IP networking principles and Internet Engineering Task Force (IETF) activities in the areas of DNS, IP Security, and Virtual Private Networks. The person of ordinary

skill is deemed to have a general knowledge of all relevant prior art including patents and published patent applications, books, academic papers, and other publications. The person of ordinary skill in the art may have at least a Bachelor's degree in engineering or computer science. The person of ordinary skill in the art may have worked in academia, for a technology company, or for a government.

III. <u>THE '151 PATENT</u>

22. For purposes of claims 1 and 13, the '151 Patent relates to automatic creation of a secure, encrypted communication channel in response to a domain-name service (DNS) look-up function (Ex.1001 at 36:58-60).

23. In the '151 Patent, a DNS request is generated from a client computer to request an IP address corresponding to a domain name that is associated with a web site hosted by a server. A determination is made whether the DNS request is requesting access to a secure web site, e.g., based on a domain name extension, or by reference to an internal table of such sites. (Ex.1001 at 37:60-65). When the DNS request corresponds to a secure web site, a VPN is automatically initiated between the client computer and the target computer. (Ex.1001 at 37:33-38). When the DNS request does not correspond to a secure web site/server, a

look-up function is performed that returns the IP address of the non-secure web site. (Ex.1001 at 37:43-48).

24. The ‘151 Patent discloses various exemplary embodiments for implementing such automatic creation of a VPN. With respect to receiving the DNS request and determining if the request corresponds to a secure web site (e.g., the user is authorized to access the secure web site), a DNS server may perform these steps. (Ex.1001 at 37:33-38). In another example, a DNS proxy may receive the DNS requests and perform the determining. (Ex.1001 at 37:60-62). In some embodiments, the DNS proxy may reside on a different machine than the DNS server (Ex.1001 at 38:33-35), and in other embodiments, the DNS proxy and DNS server may be combined in a single machine. (Ex.1001 at 38:31-33).

25. With respect to creating the encrypted channel (e.g., the VPN), the DNS server may set up the VPN between the client computer and the server. (Ex.1001 at 37:33-38). Alternatively, a DNS proxy may send a request to a gatekeeper to create the VPN between the client computer and the server. (Ex.1001 at 39:10-20). The gatekeeper facilitates the allocation and exchange of information needed to communicate securely and may send a resolved address back to the client computer via the DNS proxy. (Ex.1001 at 38:24-28, 39:10-20). In any of these examples, the VPN is established without user involvement.

26. As with the DNS proxy and DNS server, the gatekeeper and DNS server may reside on different machines or be combined on a single machine. (Ex.1001 at 38:53-55). By extension, any of the DNS proxy, DNS server, and gatekeeper may be on the same or different machines.

27. With respect to the look-up function, the DNS proxy may send a DNS request to a DNS server, which performs the look-up to return an IP address. (Ex.1001 at 38:22-24). In some embodiments, the gatekeeper instructs the DNS proxy to send the DNS request to the DNS server. (Ex.1001 at 39:28-36).

IV. <u>KIUCHI</u>

28. Based on personal experience, I can establish that the article entitled "C-HTTP – The Development of a Secure, Closed HTTP-based Network on the Internet," written by Takahiro Kiuchi and Shigekoto Kaihara (hereinafter "the Kiuchi paper" or simply "Kiuchi"), was presented to the public at the Symposium on Network and Distributed Systems Security (SNDSS) in 1996, and the paper was published in the symposium proceedings, distributed to the participants and made available to the public. At the time, I was then the Chief Scientist at SPYRUS and I gave a presentation as part of a panel discussion in session 4 at the SNDSS conference. The C-HTTP paper was presented in session 3 of the conference.

29. Similar to the '151 Patent, Kiuchi was concerned with establishing secure network links between different hosts on the Internet. (Ex.1002 at 64). In particular, the service was contemplated for use by medical institutions for protecting patient information and other private information of the institutions, although Kiuchi makes clear that the closed virtual network can be used in other areas (Ex.1002 at 69, paragraph 5). To facilitate my explanation of Kiuchi, I present the following figure (Figure 1), which shows the relevant components of Kiuchi's system (which Kiuchi calls the "C-HTTP" system):

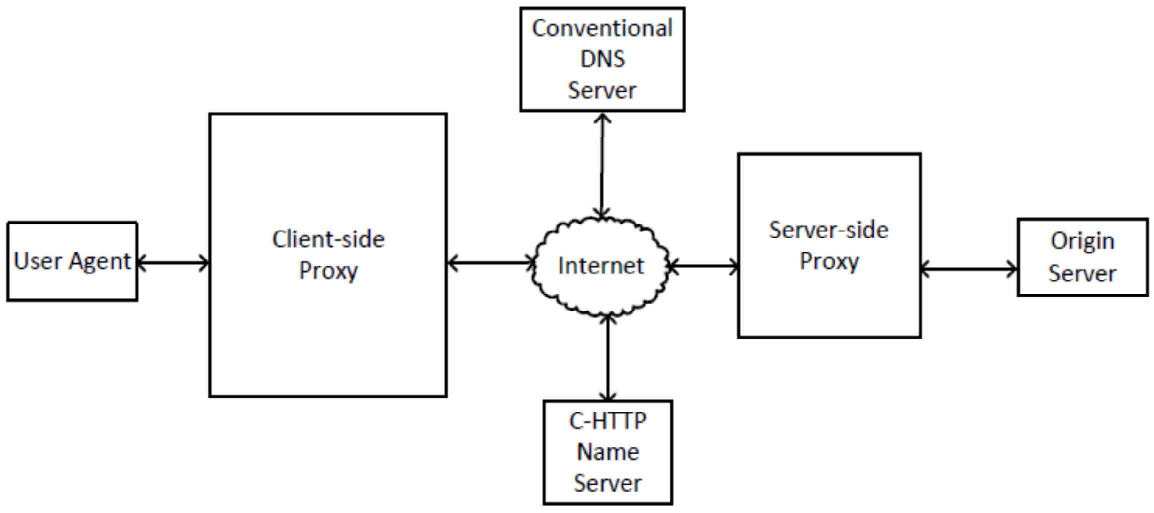


Figure 1 – Kiuchi Schematic Block Diagram

30. Kiuchi creates a closed network over the Internet that allows a user agent computer to access private web pages (HTML documents) stored on an origin server (i.e., a "secure target web site") in the closed network. The user agent

and origin server are members of the closed network constructed over the Internet using a client-side proxy that performs proxy functions for the user agent and a server-side proxy that performs proxy functions for the origin server. The client-side proxy and server-side proxy are installed in firewall devices situated between the user agent and the origin server, which are unaware of these proxies. (see Ex. 1002 at 64, sec. 2.1).

31. The user agent and the origin server are conventional HTTP/1.0 compatible devices, e.g., "[c]ommunications between two kinds of proxies and *HTTP/1.0 compatible servers/user agents within the firewalls* are performed based on HT'TP/1.0." (Ex.1002 at 64, sec. 2.1, emphasis added). It is well-known that HTTP is the communication protocol used to connect to servers on the World Wide Web. (Ex.1023 at 436). Kiuchi shows an example in which a web page (HTML document) is sent from origin server to client-side proxy (see Ex.1002 at 66, Figure (a)). The client-side proxy sends a rewritten version of the HTML document with modified links (URLs or resource names) to the user agent (see Ex.1002 at 66, Figure (b)). An end-user is able to select and request a modified link in order to generate an HTTP GET request for the requested web page (see Ex.1002 at 65, sec. 2.3(a); Ex.1002 at 66, Figure (c)(1)). Thus, it is clear that Kiuchi is providing access to private web pages at a secure target web site (i.e., the origin server).

32. The client-side proxy and server-side proxy work in conjunction with a C-HTTP name server over the Internet. (Ex.1002 at 64). For permitted secure communications, the C-HTTP name server is the server that responds to name service requests by looking up domain names and returning their IP address and related VPN resources, i.e., public key and Nonce values (Ex.1002 at 65, sec. 2.3(2)). Each proxy is registered with the C-HTTP name server, including a hostname, IP address, and public key for the proxy. (Ex.1002 at 65, sec. 2.2). The hostname (domain name) of the server-side proxy is used to access resources at an origin server being proxied by the server-side proxy. For non-secure connections, a conventional DNS name service is used to return IP addresses. *Id.* Thus, domain name services are provided by the C-HTTP name server for secure communication requests and by a conventional DNS for non-secure communication requests.

33. The client-side proxy receives, from the user agent, an HTTP request specifying a web page (HTML document) stored at the origin server and associated with a given URL. The URL in the HTTP request has the format "http://<hostname>/<web page>[connection ID]" (see the sample URL at Ex.1002 at 65, sec. 2.3(1), where "server.in.current.connection" is the hostname (i.e., "domain name") of the server-side proxy, "sample.html" is a web page stored on the origin server being proxied by the server-side proxy, and "6zdDfldfcZLj8V!i" is an optional connection ID).

34. Thus, the functions performed by Kiuchi's client-side proxy and C-HTTP name server can be represented as depicted schematically in the following figure (Figure 2):

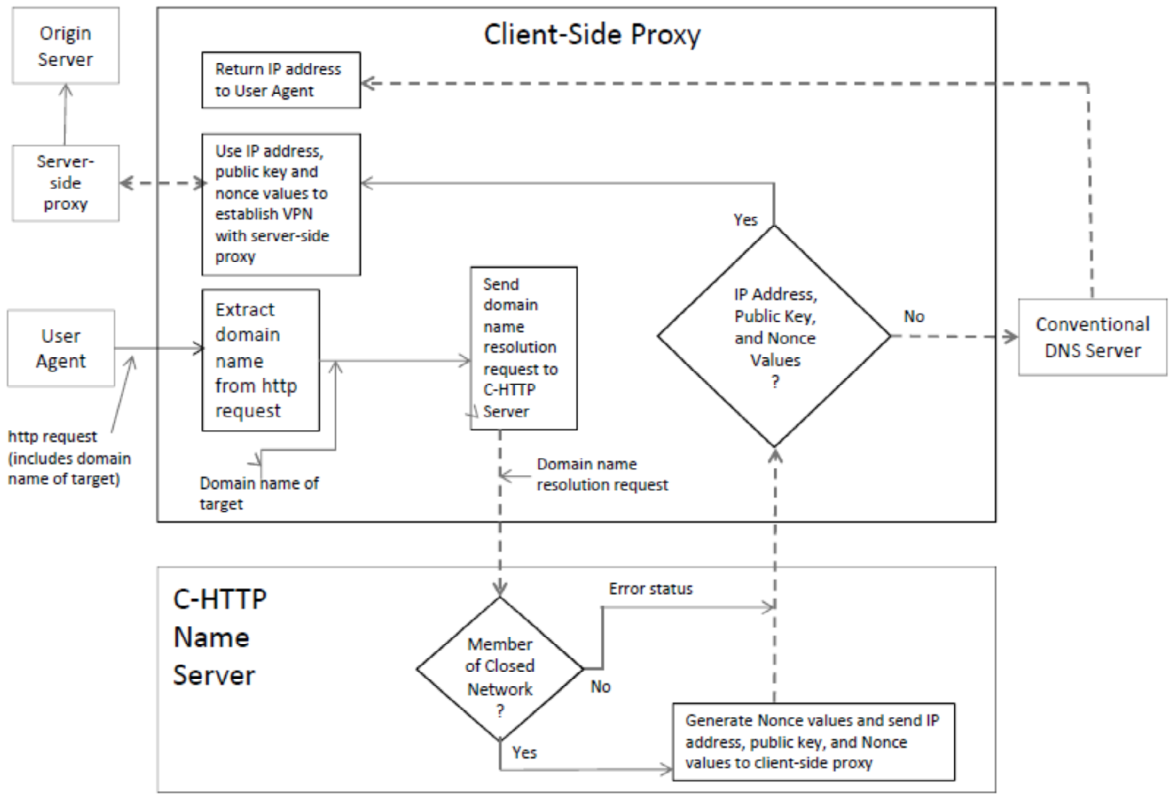

Client-Side Proxy
Origin Server
Server-side proxy
User Agent
http request (includes domain name of target)
Return IP address to User Agent
Use IP address, public key and nonce values to establish VPN with server-side proxy
Extract domain name from http request
Domain name of target
Send domain name resolution request to C-HTTP Server
Domain name resolution request
Yes
IP Address, Public Key, and Nonce Values ?
No
Conventional DNS Server
C-HTTP Name Server
Member of Closed Network ?
Error status
No
Yes
Generate Nonce values and send IP address, public key, and Nonce values to client-side proxy

Figure 2 - Functional Diagram of Kiuchi

35. In order to create a secure, encrypted channel (VPN), the client-side proxy first determines whether the HTTP request is directed to a secure server (i.e., a server-side proxy) in the closed network. Specifically, the client-side proxy "asks the C-HTTP name server whether it can communicate with the host specified in [the] URL" (Ex.1002 at 65, sec. 2.3(2)), specifically by "[taking] off the

connection ID and [forwarding] the stripped, the original resource name to the server in its request" to the name server. (Ex.1002 at 65, sec. 2.3(1)). Thus, the client-side proxy extracts the domain name from the HTTP request and sends the requested domain name to the C-HTTP name server to request the IP address of the host.

36. Upon receiving the request from the client-side proxy, the name server "examines whether the requested server-side proxy is registered in the closed network." (Ex.1002 at 65, sec. 2.3(2)). The name server returns an IP address only if the requested server-side proxy is registered in the closed network, and a secure connection with the server-side proxy is permitted. (Ex.1002 at 65, sec. 2.3(2)). Along with the IP address, the C-HTTP name server also returns the public key of the server-side proxy and Nonce values. *Id.* A Nonce value is a value used once for security/encryption. The C-HTTP name server generates and provides the Nonce values, which are later used to establish a secure connection between the client-side proxy and the server-side proxy. (see Ex. 1002 at 65, sec. 2.2). A Nonce value is used to prevent a replay attack. *Id.*

37. The client-side proxy uses the public key and Nonce values to create a secure, encrypted communication channel (i.e., VPN) with the server-side proxy (Ex.1002 at 65, sec. 2.3(3)). Specifically, the client-side proxy "sends a request for connection to the server-side proxy, which is encrypted using the server-side

proxy's public key and contains the client-side proxy's IP address, hostname, request Nonce value and symmetric data exchange key for request encryption." (Ex.1002 at 65, section 2.3(3)). Thus, the secure, encrypted channel (VPN) is automatically initiated in at least two ways, first by the C-HTTP name server, which sends the public key and Nonce values to the client-side proxy to cause creation of the VPN (which is analogous to the DNS proxy in the '151 Patent sending a message to the gatekeeper computer to request that a VPN be created – see Ex. 1001 at 37:66-38:2), then by the client-side proxy, which sends the request for connection to the server-side proxy.

38. When the server-side proxy receives the client-side proxy's IP address, the hostname and public key, it authenticates the values and generates a connection ID as well as a second key for response encryption (Ex.1002 at 65-6, sec. 2.3(4)). When these are accepted and checked by the client-side proxy, the secure, encrypted communication channel is established (Ex.1002 at 66, sec. 2.3(5)). Security between the proxies is made possible by the public key and Nonce values provided by the C-HTTP name server.

39. Once the secure, encrypted communication channel is established, HTTP/1.0 messages can then be exchanged between the user agent and the origin server over the secure, encrypted channel via the proxies (ex.1002 at 66, ¶¶ (7)-(8)).

40. If a secure connection with the requested host is <u>not</u> permitted, the name server instead returns an error status to the client-side proxy. (Ex.1002 at 65, sec. 2.3(2)). Specifically, in response to determining that the requested server-side proxy is not registered in the closed network (which indicates that the DNS request does not correspond to a secure server), the C-HTTP name server returns to the client-side proxy "a status code which indicates an error." (Ex.1002 at 65, sec. 2.3(2)). In turn, "[i]f the client-side proxy receives an error status, then it performs DNS lookup, behaving like an ordinary HTTP/1.0 proxy." (Ex.1002 at 65, sec. 2.3(2)). It is well-known that such a DNS lookup involves sending a request to a DNS server and receiving an IP address back from the DNS server. (Ex.1010 at 70 et seq.). In this way, the domain name is resolved and the IP address is returned to the client-side proxy, specifically by the client-side proxy sending a lookup request to the conventional DNS server which resolves the domain name and returns the IP address to the client-side proxy. Once the IP address is obtained, a typical non-secure communication may take place.

41. Kiuchi discloses at least two ways in which a gatekeeper computer is used for automatically initiating the VPN between the client-side proxy and the server-side proxy, one in which gatekeeper computer functions are implemented in the C-HTTP name server, and the other in which gatekeeper computer functions are implemented in the server-side proxy.

42. With regard to the former, the ‘151 Patent makes clear that the gatekeeper can be implemented as a function within the DNS server (see Ex.1001 at 38:22-24). As discussed above, the C-HTTP name server automatically initiates the VPN by sending the C-HTTP name service response to the client-side proxy. In this context, the C-HTTP name server also performs the functions of a gatekeeper computer because it allocates VPN resources, e.g., it generates and provides the request and response Nonce values and returns the public key of the server-side proxy and the request and response Nonce values to the client-side proxy. (Ex.1002 at 65, sec. 2.2).

43. With regard to the latter, as discussed above, Kiuchi’s client-side proxy automatically initiates the VPN by sending a request for connection to the server-side proxy. In this context, the server-side proxy also performs functions of a gatekeeper computer because it receives a request for connection from the client-side proxy (which is analogous to the gatekeeper computer in the ‘151 Patent receiving a message from the DNS proxy requesting that a VPN be created – Ex.1001 at 37:66-38:02) and allocates VPN resources such as a Connection ID and a second symmetric data exchange key that are used in establishing a secure connection between the client-side proxy and the server-side proxy. (see Ex.1002 at 66, sec. 2.3(4)-(5)). In order to create the VPN, the server-side proxy (i.e., the gatekeeper) has to accept the request for connection from the client-side proxy,

authenticate the client-side proxy, check the integrity of the Nonce values, and generate a connection ID and other parameters for the VPN (Ex.1002 at 65(4)-66(5)). Under the broadest reasonable interpretation, the gatekeeper computer can be a function in the target computer.

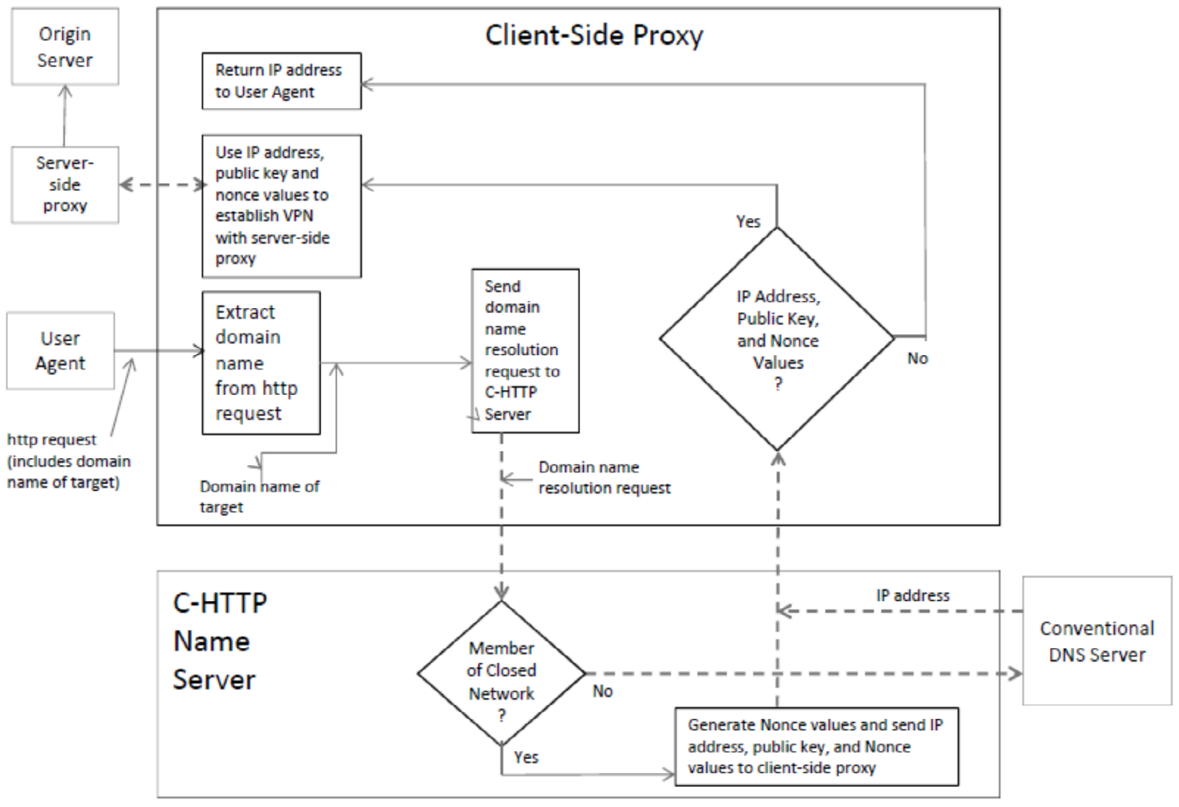
44. A person of ordinary skill in the art would have understood that devices such as the client-side proxy device, the server-side proxy device, and the C-HTTP name server device are data processing devices and that functions performed in such devices are necessarily stored in computer program code in memory, as is the case for any such processing device. A person of ordinary art also would have understood that several elements claimed in the '151 Patent – e.g., client computer, DNS proxy server, target computer, and gatekeeper computer – encompass implementation of such elements in software modules. Such software modules can reside on separate machines or be combined in ways where various functions reside on the same machine. This is the nature of software, where developers generally have great leeway in how to divide functions into software modules and where to place the software modules. Kiuchi also teaches to an ordinarily skilled artisan that the functions of a "DNS proxy server" can be included in the same machine as the client computer (i.e., the client-side proxy machine) or in a DNS server such as the C-HTTP name server, and that the

functions of a "gatekeeper computer" can be included in the server-side proxy or the C-HTTP server.

45. Moreover, it would have been a matter of design choice to a person of ordinary skill in the art to consolidate domain name resolution functions in Kiuchi's C-HTTP name server. Kiuchi clearly recognizes and discloses that a conventional DNS lookup is needed when the DNS request does not correspond to a secure server, i.e., when the requested server-side proxy is not registered in the closed network. (Ex.1002 at 65, sec. 2.3(2)). This is identical to the '151 Patent, where a DNS lookup is performed when the DNS request does not correspond to a secure server. (ex.1001 at 38:12-16).

46. Kiuchi defines three new components for the system, namely the client-side proxy, the server-side proxy, and the C-HTTP name server. (Ex.1002 at 64, sec. 2.1). While Kiuchi describes a system in which a conventional DNS lookup request is made from the client-side proxy, it would have been apparent to a person of ordinary skill in the art based on Kiuchi's teachings to make the conventional DNS lookup request from the C-HTTP name server. As discussed above, the C-HTTP name server already determines whether the DNS request received from the client-side proxy corresponds to a secure server in the closed network. Rather than returning an error status to the client-side proxy when the DNS request does not correspond to a secure server, it would have been trivial and

obvious as a mere design choice for the C-HTTP name server to forward the DNS request to the conventional DNS server by passing the domain name received in the C-HTTP name service request to the conventional DNS server, as depicted in the following figure (Figure 3):



Origin Server
Client-Side Proxy
Return IP address to User Agent
Server-side proxy
Use IP address, public key and nonce values to establish VPN with server-side proxy
Yes
Send domain name resolution request to C-HTTP Server
IP Address, Public Key, and Nonce Values ?
No
User Agent
Extract domain name from http request
http request (includes domain name of target)
Domain name of target
Domain name resolution request
C-HTTP Name Server
IP address
Conventional DNS Server
Member of Closed Network ?
No
Yes
Generate Nonce values and send IP address, public key, and Nonce values to client-side proxy

Figure 3 – Alternate Functional Diagram of Kiuchi

47. Such a configuration, which places a DNS proxy server function in a modified C-HTTP name server, is merely a rearrangement of existing functions within the C-HTTP system and could be implemented with or without modifying Kiuchi's protocols. For example, a C-HTTP name service response message

containing an IP address without a public key and Nonce values (e.g., using values of zero or other convention for the public key and Nonce fields, or modifying the protocol to use a previously unused flag in the response to indicate that a public key and Nonce values are not provided) would indicate to the client-side proxy that the DNS request does not correspond to a secure server and hence that no VPN is needed. The motivation for modifying Kiuchi in this way would have been to streamline the operation of the system, e.g., instead of having the C-HTTP name server send an error status to the client-proxy which would in turn initiate a conventional DNS inquiry, the modification eliminates the error status message from the process by having the C-HTTP name server directly initiate the request to the conventional DNS server.

48. I have gone through the claims in view of Kiuchi as discussed in paragraphs 29-47 above and have set forth the correspondence between them, element by element, as set forth in the claim chart attached as Appendix C.

49. Additionally, Kiuchi's client-side proxy performs a "resolver" function that receives a domain name resolution request from an internal client (i.e., the domain name extracted from the received HTTP request) and returns an IP address for the domain name. The client and resolver functions performed by Kiuchi's client-side proxy can be represented as depicted schematically in the following figure (Figure 4):

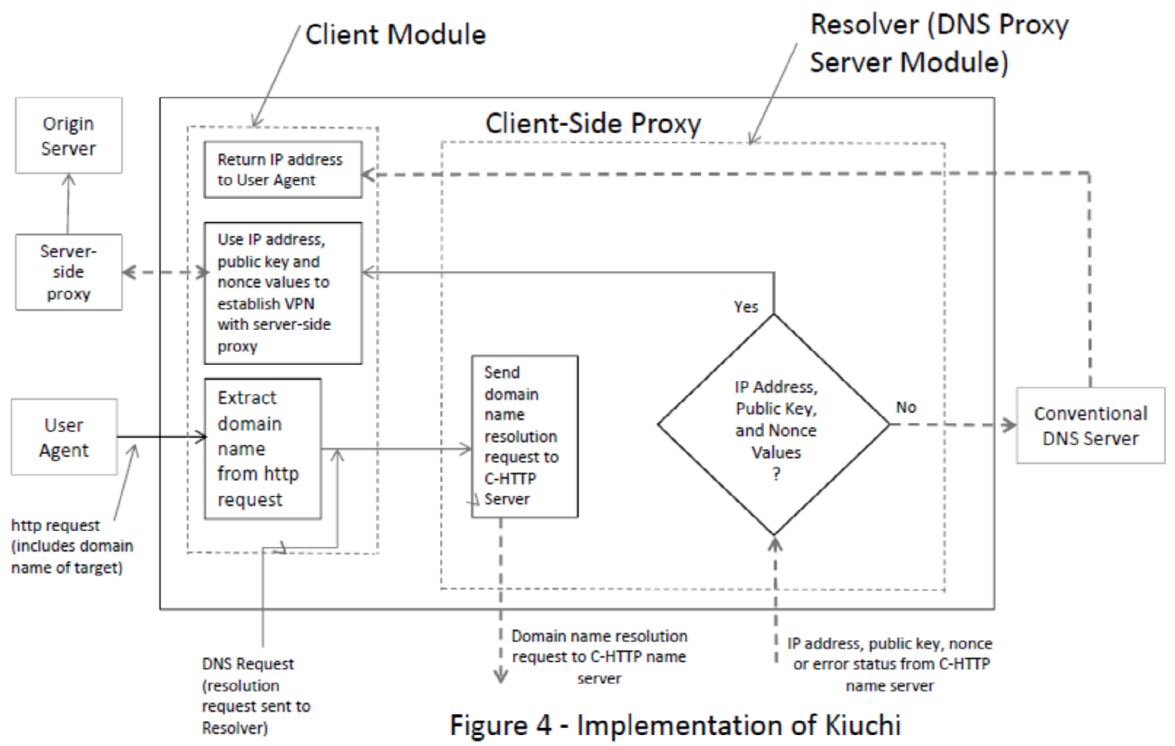

Client Module
Resolver (DNS Proxy Server Module)
Client-Side Proxy
Origin Server
Server-side proxy
User Agent
http request (includes domain name of target)
Return IP address to User Agent
Use IP address, public key and nonce values to establish VPN with server-side proxy
Extract domain name from http request
Send domain name resolution request to C-HTTP Server
IP Address, Public Key, and Nonce Values ?
Yes
No
Conventional DNS Server
DNS Request (resolution request sent to Resolver)
Domain name resolution request to C-HTTP name server
IP address, public key, nonce or error status from C-HTTP name server

Figure 4 - Implementation of Kiuchi

50. Prior to February 2000, it was well-known for a client function of a client computer (e.g., an application such as a web browser) to request domain name resolution from a resolver function in the client computer. For example, client applications running on Windows and Unix operating systems made function calls to the operating system (specifically, the "gethostbyname" function) in order to obtain an IP address for a given hostname (see Ex.1004 – "From an application's point of view, access to the DNS is through a resolver …. The [gethostbyname(3) library function] takes a hostname and returns an IP address…The resolver contacts one or more name servers to do the mapping"). Ex.1005 at 112 describes error return codes from gethostbyname() and

gethostbyaddr() library functions “when using the resolver.” Ex.1006 describes the gethostbyname eCos system library function. Ex.1007 at 2:63-3:8 states that "When a user program, such as the browser, requests information ... a resolution request is passed in the form of a query to the resolver." Ex.1008 at 9:49-54 states that "Access to the DNS is through a resolver and software library functions. The function in this case takes a domain name or host name and returns an IP address."

51. Thus, in Kiuchi, an internal resolution request is made to the resolver function (which is a collection of software functions within the client-side proxy), which acts as a DNS proxy server to contact the C-HTTP name server and optionally also the conventional DNS server to obtain an IP address for the domain name and return the IP address to the internal client. This internal resolution request is a domain name service request because it is a communication that contains a domain name and requests an IP address for the domain name.

52. Furthermore, a careful consideration of the inner workings of the client-side proxy also reveals that the resolver function performs functions that map directly to the functions performed by the DNS proxy server of the ‘151 Patent. The DNS proxy server of the ‘151 Patent receives a DNS request, determines whether access to a secure web site has been requested (e.g., based on a domain name extension or by reference to an internal table of such sites), automatically initiates a VPN if access to a secure target web site has been

requested, and passes through the DNS request to a conventional DNS server if access to a non-secure site had been requested (see Ex.1001 at 37:60-38:16). Similarly, the resolver function of Kiuchi receives a DNS request, determines whether the DNS request corresponds to a secure server (based on a query to the C-HTTP name server, which essentially is just a remote table lookup similar to the internal table lookup of the '151 Patent), automatically initiates/creates a secure, encrypted channel (VPN) if access to a secure target web site has been requested, and forwards the DNS request to a conventional DNS server if the DNS request does not correspond to a secure server.

53. Thus, when the Client Module receives the HTTP request from the user agent, it extracts the hostname from the URL received in the HTTP request and sends an internal resolver request containing the domain name to the DNS Proxy Server Module. This internal resolver request is a "DNS request" because it is a communication that contains a domain name (i.e., the hostname from the URL in the HTTP request) and requests an IP address for the domain name.

54. Upon receiving the DNS request from the Client Module, the DNS Proxy Server Module sends the domain name to the C-HTTP name server in the form of a C-HTTP name service request to ask the C-HTTP name server whether the client-side proxy can communicate with the specified host. (Ex.1002 at 65, sec. 2.3(2)).

55. The C-HTTP name server "examines whether the requested server-side proxy is registered in the closed network." (Ex.1002 at 65, sec. 2.3(2)). The C-HTTP name server determines whether the DNS request transmitted by the Client Module in step (1) is requesting access to a server-side proxy in the closed network based on whether the requested server-side proxy is registered in the closed network. The C-HTTP name server sends a C-HTTP name service response to the DNS Proxy Server Module containing "the IP address and public key of the server-side proxy and both request and response Nonce values" (Ex.1002 at 65, sec. 2.3(2)), if the requested server-side proxy is registered in the closed network. The C-HTTP name server sends "a status code which indicates an error," if the requested server-side proxy is not registered in the closed network. (Ex.1002 at 65, sec. 2.3(2)-(3)).

56. Additionally, the DNS Proxy Server Module determines whether the DNS request sent by the Client Module corresponds to a server-side proxy in the closed network and thus determines whether the DNS request corresponds to a secure server based on the type of response received from the C-HTTP name server. In particular, the client-side proxy determines that the DNS request corresponds to a secure server, only if a C-HTTP name service response is returned, and determines that the DNS request does not correspond to a secure server, if the response is an error status status.

57. If the DNS Proxy Server Module receives a C-HTTP name service response from the C-HTTP name server and therefore determines that the DNS request corresponds to a secure server, the DNS Proxy Server Module automatically initiates/creates a secure, encrypted channel (VPN), specifically by the DNS Proxy Server Module sending the IP address and VPN resources (e.g., the public key and Nonce values) to the Client Module and the Client Module "[sending] a request for connection to the server-side proxy, which is encrypted using the server-side proxy's public key and contains the client-side proxy's IP address, hostname, request Nonce value and symmetric data exchange key for request encryption." (Ex.1002 at 65, right column, section 2.3(3)). In this regard, it should be noted that the '151 Patent provides, as one example of automatically initiating/creating a secure, encrypted channel, transmission of a message requesting that a VPN be created (see Ex.1001 at 37:66-38:2). Sending the IP address and VPN resources by the DNS Proxy Server Module to the Client Module is analogous because it is a message that causes the secure, encrypted channel to be created.

58. Additionally or alternatively, the Client Module automatically initiates the VPN in response to receiving the IP address, public key, and Nonce values, specifically by "[sending] a request for connection to the server-side proxy, which is encrypted using the server-side proxy's public key and contains the client-side

proxy's IP address, hostname, request Nonce value and symmetric data exchange key for request encryption." (Ex.1002 at 65, section 2.3(3)).

59. In response to determining that the DNS request does not correspond to a secure server, the DNS Proxy Server Module performs a DNS lookup as "an ordinary HTTP/1.0 proxy" (Ex.1002 at 65, section 2.3(2)) and returns an IP address to the Client Module. It is well-known that such a DNS lookup involves sending a request to a DNS server and receiving an IP address back from the DNS server. (Ex.1010 at 70 et seq.). Thus, the DNS Proxy Server Module passes the domain name generated by the Client Module to the conventional DNS server and hence forwards the request to the conventional DNS server in order to perform the DNS lookup. In this way, the domain name is resolved and the IP address is returned to the client computer (i.e., Client Module), specifically by the DNS Proxy Server Module performing the DNS lookup (through a request to a conventional DNS) and returning the IP address to the Client Module.

60. As discussed above, during creation of the secure, encrypted channel (VPN), the server-side proxy performs functions of a gatekeeper computer because it allocates VPN resources, such as a Connection ID and a second symmetric data exchange key, that are used in establishing a secure connection between the client-side proxy and the server-side proxy. (Ex.1002 at 66, section 2.3(5)).

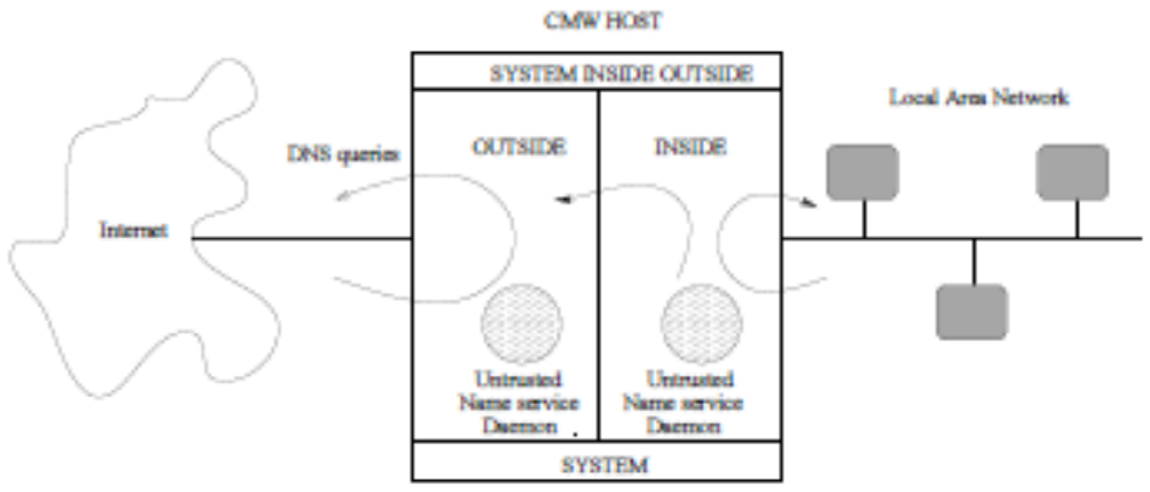
61. I have gone through the claims in view of Kiuchi as discussed in paragraphs 49-60 above and have set forth the correspondence between them, element by element, as set forth in the claim chart attached as Appendix D.

V. **DALTON/KIUCHI**

62. Prior to October 1998, use of the Internet was expanding exponentially. Organizations with multiple locations were moving away from costly private circuits for local communication to closed virtual private networks implemented on the Internet. Kiuchi is just one of many references describing how to implement a closed network on the Internet. The '151 patent concedes that in the prior art, a "tremendous variety of methods have been proposed and implemented to provide security and anonymity for communications over the Internet." (Ex. 1001, 1:27-29). The cost incentive to set up secure connections over the Internet instead of resorting to private leased circuits was a major factor in the shift to reliance on the Internet.

63. Dalton described a firewalled Domain Name System in a single machine, referred to as a Compartmented Mode Workstation (CMW). (Ex.1003 at 711-1). The CMW provides a DNS service (i.e., a CMW-based DNS service – Ex.1003 at 711-3) such that clients on an internal (closed) network can have access

to both public and private hosts on the internal network as well as access to hosts on an external network, while hosts on the external network can have access to only public hosts on the internal network. Dalton makes clear that the Domain Name System (DNS) used in the CMW-based DNS service is the conventional DNS defined by the IETF. (see Ex.1003 at 711-4, referencing IETF Request For Comments (RFC) 1034 and 1035 – Dalton references [5,6]). Also, aside from the fact that it is well-known that DNS queries request an IP address associated with a domain name, Dalton discloses that "DNS provides a mapping between host names [i.e., domain names] and numerical Internet Protocol addresses." (Ex.1003 at 711-3). Dalton illustrates a simple example of the CMW-based system having two zones (i.e., internal/protected and external/public) as follows (Ex.1003 at 711-4, Figure 2):



64. A person of ordinary skill in the art would have understood that the CMW is a data processing device and that functions performed in the CMW are necessarily stored in computer program code in memory, as is the case for any such processing device. Generally speaking, the CMW intercepts and processes DNS requests from external clients on the public Internet and internal clients on the closed, private Local Area Network (LAN). The LAN is protected against unauthorized access from external clients. Specifically, external clients are only permitted to access specific servers on the LAN. However, internal clients are permitted to access all servers on the LAN as well as servers on the Internet. When the CMW receives a DNS request from an internal client on the LAN, it determines whether the DNS request is requesting access to an internal host on the LAN based on a set of DNS records maintained in the CMW. If the DNS request is requesting access to an internal host on the LAN, then the CMW can resolve the IP address locally, as represented by the arrow looping back to the LAN in the above figure. However, if the DNS request is requesting access to an external host on the Internet, then the CMW forwards the DNS request to a DNS server on the Internet, as represented by the arrow from the name service daemon in the "INSIDE" box to the "OUTSIDE" box in the above figure. Dalton makes clear that an internal host can be a World Wide Web (web) server (Ex.1003 at 711-3)

and that a client can include a browser (Ex.1003 at 711-6). Thus, Dalton teaches secure access to web sites by an internal client to an internal host.

65. In order to implement this security arrangement, the CMW runs two name service daemons, one for the external/public zone (i.e., SYSTEM OUTSIDE) and one for the internal/protected zone (i.e., SYSTEM INSIDE). (Ex.1003 at 711-3). Specifically, "The name service daemon running at the level SYSTEM OUTSIDE has the minimal subset of DNS records, consisting of only those necessary for external systems to locate locally provided public services such as World Wide Web servers and mail gateways. Conversely, the name service daemon running at the SYSTEM INSIDE level holds a full set of DNS data for that zone. Internal querying clients therefore have access to all the DNS records for their zone but external clients have a much more limited view." (Ex.1003 at 711-3, 4). "The aim of the CMW-based DNS service is to provide different DNS information to querying clients depending upon whether the query originated from the external network or the internal network." (Ex.1003 at 711-3). Dalton states that "External clients should have access only to DNS records stored at SYSTEM OUTSIDE. Internal clients should have access to records stored at SYSTEM INSIDE. Additionally, the internal querying clients should be allowed access to DNS information from name servers on the external network. External clients should not be able to access internal DNS information." (Ex.1003 at 711-3).

66. Internal clients on the LAN can access local servers on the LAN or external servers on the Internet. To accomplish this, the CMW includes a front end daemon that intercepts DNS requests and forwards answers back to the querying clients (i.e., the front end daemon "sits and waits for incoming DNS queries (both TCP and UDP) and for answers sent back from the untrusted name service daemons that require forwarding over the network to the original querying clients"). (Ex.1003 at 711-4). When a DNS packet is intercepted by the front end daemon from an internal client, the front end daemon simply forwards it to the name service daemon running at the SYSTEM INSIDE level ("When a DNS packet is received, it is simply forwarded by the front end daemon to the particular non-privileged name service daemon running at the sensitivity level of the incoming packet"). (Ex.1003 at 711-4). The name service daemon is responsible for resolving the DNS request and may contact other name servers, such as the DNS server on the Internet, in order to resolve the IP address. Specifically, each name service daemon is "responsible for doing the actual resolving of the [DNS] query" made to the CMW for its respective zone. (Ex.1003 at 711-4). "Any communication [a name service daemon] wishes to carry out on the network, such as contacting other name servers, must be proxied through the trusted front end wrapper daemon." (Ex.1003 at 711-4).

67. In the simple two-zone example, "the front end daemon would be configured to proxy packets for the SYSTEM OUTSIDE name service daemon over the SYSTEM OUTSIDE network only and to proxy packets at the request of the SYSTEM INSIDE name service daemon over either the SYSTEM INSIDE or SYSTEM OUTSIDE networks. This allows the internal name server to query other non-local name servers out on the Internet in order to resolve an external address for an internal client but blocks external clients from accessing the internal name servers." (Ex.1003 at 711-4).

68. Each name service daemon maintains DNS records for its respective zone, and when an internal DNS client sends a DNS request for an internal host, the front end daemon passes the DNS request to the name service daemon at the SYSTEM INSIDE level, which "holds a full set of DNS data for that zone." (Ex.1003 at 711-3). Each name service daemon is name service daemon is "responsible for doing the actual resolving of the [DNS] query." Thus, if the name service daemon has a DNS record for the DNS request (indicating that the requested host is on the inside network), then the name service daemon can resolve the IP address locally and pass the IP address back to the front end daemon to be forwarded back over the inside network to the original querying client, as represented by the arrow looping back to the LAN in the above figure.

69. However, when the internal DNS client sends a DNS request for an external host such that the name service daemon needs to query a non-local name server on the Internet in order to resolve the client query, it sends the DNS request to the external DNS server on the Internet, specifically by sending a query packet to an associated external proxy agent, which simply "sends the packet out over the network, and passes any replies it receives back to the untrusted name service daemon" to be passed back to the original querying client via the front end daemon. (Ex.1003 at 711-5).

70. Dalton does not teach automatic initiation of a VPN between the internal DNS client and the requested internal host computer. However, it was well-known for a domain name server or DNS proxy server to return VPN resources along with an IP address in response to a DNS request that is requesting access to a secure target web site and for a VPN to be automatically initiated based on those VPN resources. For example, Kiuchi teaches a DNS server (i.e., the C-HTTP name server) that receives a DNS request and, in response to determining that the DNS request is requesting access to a secure target web site, returns an IP address along with VPN resources (e.g., a public key and Nonce values) that can be used for automatically initiating a VPN between client and target computers. (Ex.1002 at 65, sec. 2.3(2)). The client-side proxy in turn sends a request for a

secure connection to the server-side proxy using the IP address, the public key, and the Nonce values. (Ex.1002 at 65, sec. 2.3(3)).

71. As discussed in Kiuchi (see Ex.1002 at 69), there are many advantages to replacing a private, closed network of the type used in Dalton (i.e., the Local Area Network) with a closed virtual network constructed on the Internet, e.g., convenience, speed, and cost. Thus, Kiuchi expressly provides motivation to replace Dalton's closed, private network with a virtual private network of the type taught by Kiuchi so that a VPN is automatically initiated between an "internal" client computer and an "internal" target computer in order to maintain secure communications between the two computers as if the two computers were still operating on the closed, private network.

72. Replacing Dalton's private, closed LAN with a closed virtual network of the type taught by Kiuchi would involve merely modifying Dalton's CMW to perform Kiuchi's name server functions (i.e., to return an IP address along with VPN resources if the DNS request from the "internal" client computer is requesting access to an "internal" target computer) and connecting the internal hosts to the Internet using the CMW as their DNS (proxy) server. It would have been apparent to a person of ordinary skill to modify Dalton's CMW in this way because the Dalton's CMW and Kiuchi's name server are both DNS servers that perform a lookup service and return an IP address for a requested domain name

(Dalton's CMW is additionally or alternatively a DNS proxy server because it responds to a domain name inquiry in place of a DNS). Modification of Dalton also would involve adding VPN establishment functions performed by Kiuchi's client-side proxy and server-side proxy (e.g., implemented respectively in the client and target computers, or implemented respectively in firewall devices protecting the client and target computers as in Kiuchi) to automatically initiate a VPN based on the VPN resources returned by the modified CMW. Such a modified CMW also would handle requests from the server-side VPN establishment function to authenticate the client computer as in Kiuchi (see Ex.1002 at 65-66).

73. I have gone through the claims in view of the combination of Dalton and Kiuchi and have set forth the correspondence between them, element by element, as set forth in the claim chart attached as Appendix E.

TABLE OF APPENDICES

Appendix	Description
A.	Curriculum Vitae (CV) of Russell Housley
B.	List of Materials Considered
C.	First Invalidity Claim Chart in view of Kiuchi
D.	Second Invalidity Claim Chart in view of Kiuchi
E.	Invalidity Claim Chart in view of Dalton and Kiuchi

APPENDIX A

RUSSELL HOUSLEY

TECHNICAL SUMMARY

Russell Housley is currently serving as Chair of the Internet Engineering Task Force (IETF). He is co-author of *Planning for PKI* published by John Wiley & Sons. He was a key contributor to the Secure Data Network System (SDNS) protocol development group contributing to protocols for secure communications, certificate management, and keying material distribution. He co-authored the SDNS Security Protocol 4 (SP4) and Message Security Protocol (MSP). He also co-authored other security protocols such as the Secure Data Exchange Protocol (SDE) and IEEE 802.10c Key Management Protocol. He is editor and key contributor for the IETF Public Key Infrastructure using X.509 (PKIX) working group. PKIX is defining the Internet Public Key Infrastructure (PKI). He provides technical advice to the Department of Commerce and the Department of Defense on PKI and Key Management Infrastructure (KMI). Over the last ten years, he has performed security and vulnerability analyses of various communications architectures and security policies based on known threats and proposed certification criteria.

EDUCATION

M.S. Computer Science, George Mason University, Fairfax, VA	**1992**
B.S. Computer Science, Virginia Polytechnic Institute and State University, Blacksburg, VA	**1982**

PATENTS

US Patent 6,003,135:	Modular security device
US Patent 6,088,802:	Peripheral device with integrated security functionality
US Patent 6,904,523:	Method and system for enforcing access to a computing resource using a licensing attribute certificate
US Patent 6,981,149:	Secure, easy and/or irreversible customization of cryptographic device
US Patent 7,356,692:	Method and system for enforcing access to a computing resource using a licensing attribute certificate

EMPLOYMENT HISTORY

Vigil Security, LLC, <u>Founder and Owner</u> **2002 - Present**

Beginning in March 2013, serving as Chair of the Internet Architecture Board (IAB), and beginning May 2013, serving as a member of the Internet Research Steering Group (IRSG).

From March 2007 to March 2013, served as Chair of the Internet Engineering Task Force (IETF). Managed the open and transparent technical standards process for the Internet.

From March 2003 to March 2007, served as the IETF Security Area Director. Provided leadership to many working groups that were developing security standards for the Internet, including the Public Key Infrastructure using X.509 (PKIX), IP Security (IPsec), Transport Layer Security (TLS), Secure MIME (S/MIME), Domain Keys Identified Mail (DKIM), Long-Term Archive and Notary Services (LTANS), and Multicast Security (MSEC) working groups.

From 2001 until March 2007, contributed to Wireless Local Area Network (WLAN) security solution development in the IEEE 802.11 working group.

Provide consulting on security protocols, security architectures, and Public Key Infrastructure (PKI).

Chair of CertiPath Policy Management Authority: Cross certifying with this Bridge Certification Authority for the defense and aerospace industry demonstrates operational excellence in identity management and security practices. Assisted with the transition from SHA-1 to SHA-256.

WiMAX Forum Policy Authority: Provide technical and policy advice to the WiMAX Forum for the PKI that is used to authenticate a WiMAX Device and the separate PKI that is used to authenticate the AAA server within a WiMAX network.

Consultant to U.S. Government: Helping with Crypto Modernization activities, especially in the areas of secure firmware loading, trust anchor management, public key infrastructure, and key management infrastructure.

Advisor: Member of the Advisory Board for the Georgetown Center for Secure Communications (GCSC) at Georgetown University, the Security and Software Engineering Research Center (S2ERC) at Georgetown University, and the Center for Information Assurance at the University of Dallas, Graduate School of Management. Technical advisor to Penango.

RSA Laboratories, <u>Senior Consulting Architect</u> **2001 - 2002**

Conducted research in PKI, security protocols, and implementation assurance.

Chairman of the Internet Engineering Task Force (IETF) S/MIME Working Group, and author of the Cryptographic Message Syntax (CMS). Author of RFC 3369 and RFC 3370, which update previous work on RFC 2630.

Editor and key contributor for the IETF Public Key Infrastructure using X.509 (PKIX) working group. Co-author of RFC 3279, RFC 3280, and RFC 3281, which updates and extends previous work on RFC 2459. Co-author of RFC 3379, which defines requirements for a protocol to provide delegated path validation.

Helped develop standards for short-term and long-term security solutions for IEEE 802.11 Wireless Local Area Network (WLANs). Previous effort, called WEP, has major flaws. The short-term solution must operate on the fielded hardware, but the long-term solution allowed for new hardware development. Co-inventor of the Counter with CBC-MAC (CCM) cryptographic mode, which provides both authentication and confidentiality using a single key.

Advisor to U.S. Government. Member of the President's Export Council Subcommittee on Encryption (PECSENC). Technical advisor to the National Institute for Standards and Technology (NIST) on Public Key Infrastructure (PKI) and Key Management Infrastructure (KMI).

Developed a plan to improve the assurance of all RSA Security product development efforts.

SPYRUS, <u>Chief Scientist</u> **1994 - 2001**

Responsible for technical direction of SPYRUS cryptographic token products, PKI products, and standards strategy.

Developed electronic mail security protocol meeting the needs for industry, government, and military users by combining capabilities of MSP and S/MIME. Chairman of the Internet Engineering Task Force (IETF) S/MIME Working Group, and author of the Cryptographic Message Syntax (CMS), RFC 2630. Co-author of SDNS Message Security Protocol Revision 4.0, and editor of the companion ACP 120 specification, the Common Security Protocol.

As editor and key contributor for the IETF Public Key Infrastructure using X.509 (PKIX) working group, making significant contributions to the definition of the Internet Public Key Infrastructure (PKI). Co-author of RFC 2459, RFC 2528, and RFC 2585.

Prior to completing their work in 1999, Vice Chair of the IEEE LAN/MAN Security Working Group (IEEE 802.10). Co-author and editor of the Key Management specification (IEEE 802.10c) and major contributor to the IEEE 802.10a and 802.10e Local Area Network security standards.

Contributor to American National Standards Institute (ANSI) standards for the security of financial systems. Specializing in the areas of certificate management, key management, and cryptography, contributed to the X9.41. X9.42, X9.52, X9.55, and X9.57 standards.

Advisor to U.S. Government. Member of the President's Export Council Subcommittee on Encryption (PECSENC). Technical advisor to the National Institute for Standards and Technology (NIST) on Public Key Infrastructure (PKI) and Key Management Infrastructure (KMI).

Served on the program committee and then the steering committee for the Network and Distributed System Security (NDSS) conference.

Vista Laboratory, Manager, Information Security Projects **1982-1994**

As a member of the SDNS protocol design team, helped design communications protocols for secure communication and over-the-air rekey. Co-author of Security Protocol 4 (SP4) and Message Security Protocol (MSP). Made significant contributions to the SDNS Key Management Protocol (KMP).

As a member of the SDNS INFOSEC Working Group, performed security and vulnerability analysis of the Defense Messaging System architecture and security policy based on known threats and proposed certification criteria.

Co-chair of the IEEE LAN/MAN Security Working Group (IEEE 802.10). Served on the IEEE Project 802 Executive Committee. Co-author of the Secure Data Exchange protocol (IEEE 802.10b).

Program manager and chief architect for the Trusted Xerox Network System. Responsible for the system design, system implementation, and coordination of the National Computer Security Center evaluation. Responsible for an annual budget of $1.2M.

Provided technical support for the Xerox Encryption Unit (XEU) and designed XEU enhancements. Designed and developed the Xerox Ethernet Tunnel (XET). The XET enables the XEU to be used in WANs and LANs. The tunneling protocol used in the XET was later published in RFC 3378.

As a member of the Privacy and Security Research Group (PSRG), helped design the privacy-enhanced electronic mail (PEM) system, including a certificate-based key management scheme. Also, helped start the Network and Distributed System Security (NDSS) conference, which has become an annual event sponsored by the Internet Society. Served on the first program committee, and then co-chair of the program committee in 1994.

US Air Force, <u>Systems Programmer/Analyst</u> **1982-1986**

Responsible for security and communications on five Honeywell Multics systems. Performed security audits on operating system modifications. Installed multilevel secure LAN between four systems using a Network Systems Corporation HYPERchannel and TCP/IP protocols.

Self-employed, <u>Scientific Programming Consultant</u> **1981-1982**

Designed, implemented, maintained, and documented interactive graphics software as part of a geographic data display system called TACK. Designed general graphics terminal and database interfaces. Development and testing was performed using an interactive graphics test bed that was used to gather graphics requirements for CAMS II.

PUBLICATIONS

Books

Housley, Russ, and Tim Polk. *Planning for PKI – Best Practices Guide for Deploying Public Key Infrastructure*. New York: John Wiley & Sons, 2001.

Turner, Sean and Russ Housley. *Implementing Email and Security Tokens: Current Standards, Tools, and Practices*. New York: John Wiley & Sons, 2008.

Papers

Branstad, Dennis, Joy Dorman, Russell Housley, and James Randall. "SP4: A Transport Encapsulation Security Protocol." In *Tenth National Computer Security Conference Proceedings*, September 1987, pp 158-161.

Branstad, Dennis, Joy Dorman, Russell Housley, and James Randall. "SP4: A Transport Encapsulation Security Protocol." In *Third Aerospace Security Conference Proceedings*, December 1987. [Revision of earlier work.]

Housley, Russell. "Encapsulation Security Protocol Design for Local Area Networks." In *Local Area Network Security: Workshop LANSEC '89 Proceedings*, April 1989, pp 103-109.

Migues, Sammy, and Russell Housley. "Designing a Trusted Client-Server Distributed Network." In *Fifth Annual Computer Security Applications Conference Proceedings*, December 1989, pp 91-94.

Housley, Russell. "Authentication, Confidentiality, and Integrity Extensions to the XNS Protocol Suite." *SIGSAC Review*, ACM Press, Fall 1989, pp 17-24.

Housley, Russell. "Electronic Message Security: A Comparison of Three Approaches." In.*Fifth Annual Computer Security Applications Conference Proceedings*, December 1989, pp 29.

Housley, Russell. "Security Labels in Open Systems: A Position Paper." In *Security Labels for Open Systems - An Invitational Workshop*, June 1990, NISTIR 4362, pp 83-84.

Migues, Sammy, and Russell Housley. "A Security Policy for Trusted Client-Server Distributed Networks." In *Thirteenth National Computer Security Conference Proceedings*, October 1990, pp 237-242.

Housley, Russell. "Security Labels in Open Systems Interconnection." In *Thirteenth National Computer Security Conference Proceedings*, October 1990, pp 37-43.

Housley, Russell and Sammy Migues. "Comments on the draft FIPS - Standard Security Label for the Government Open Systems Interconnection Profile." In *Standard Security Label for GOSIP - An Invitational Workshop*, June 1991, NISTIR 4614, pp 105-106.

Housley, Russell. "Security Label Framework for the Internet." RFC 1457, May 1993.

Housley, Russell. "Message Security Protocol." *InSight*, May 1995, pp 12.

Zmuda, James E., and Russell Housley. "Experiences with implementing messaging security in MSMail 3.2." In *Eighteenth National Information System Security Conference Proceedings*, October 1995, pp 281-290.

Housley, Russell and Jan Dolphin. "Metering: A Pre-pay Technique." Photonics '97 Proceedings, February 1997, SPIE vol. 3022, pp. 527-531.

Housley, Russell, Warwick Ford, David Solo, and Tim Polk. "Internet X.509 Public Key Infrastructure, Certificate and CRL Profile." RFC 2459, January 1999.

Housley, Russell, and Tim Polk. "Internet X.509 Public Key Infrastructure, Representation of Key Exchange Algorithm (KEA) Keys in Internet X.509 Public Key Infrastructure Certificates." RFC 2528, March 1999.

Housley, Russell, and Paul Hoffman. "Internet X.509 Public Key Infrastructure, Operational Protocols: FTP and HTTP." RFC 2585, May 1999.

Housley, Russell. "Cryptographic Message Syntax." RFC 2630, June 1999.

Arsenault, Al, and Russell Housley. "Protection Profiles for Certificate Issuing and Management Systems." In *22nd National Information Systems Security Conference Proceedings*, October 1999, pp 189-199.

Housley, Russell, Peter Yee, and William Nace. "[File Transfer Protocol] Encryption using KEA and SKIPJACK." RFC 2773, February 2000.

Housley, Russell. "Key Agreement for Secure Electronic Mail: Ephemeral-Static Diffie-Hellman." In *Twelfth Annual Canadian Information Technology Security Symposium Proceedings*, June 2000, pp. 185-193.

Housley, Russell, Todd Horting, and Peter Yee. "TELNET Authentication Using DSA." RFC 2943, September 2000.

Housley, Russell, Todd Horting, and Peter Yee. "TELNET Authentication Using KEA and SKIPJACK." RFC 2951, September 2000.

Housley, Russell. "Triple-DES and RC2 Key Wrapping." RFC 3217, December 2001.

Polk, William, Russell Housley, and Larry Bassham. "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile." RFC 3279, April 2002.

Housley, Russell, William Polk, Warwick Ford, and David Solo. "Internet X.509 Public Key Infrastructure: Certificate and Certificate Revocation List (CRL) Profile." RFC 3280, April 2002.

Farrell, Stephen, and Russell Housley. "An Internet Attribute Certificate Profile for Authorization." RFC 3281, April 2002.

Housley, Russell. "Cryptographic Message Syntax (CMS)." RFC 3369, August 2002.

Housley, Russell. "Cryptographic Message Syntax (CMS) Algorithms." RFC 3370, August 2002.

Housley, Russell, and Scott Hollenbeck. "EtherIP: Tunneling Ethernet Frames in IP Datagrams." RFC 3378, September 2002.

Pinkas, Denis, and Russell Housley. "Delegated Path Validation and Delegated Path Discovery Protocol Requirements." RFC 3379, September 2002.

Schaad, James, and Russell Housley. "Advanced Encryption Standard (AES) Key Wrap Algorithm." RFC 3394, September 2002.

Schaad, James, and Russell Housley. "Wrapping a Hashed Message Authentication Code (HMAC) key with a Triple-Data Encryption Standard (DES) Key or an Advanced Encryption Standard (AES) Key." RFC 3537,
May 2003.

Housley, Russ, and William Arbaugh. "Security Problems in 802.11-based Networks." Communications of the ACM, Vol 46, No 5 (May 2003): 31-34.

Cam-Winget, Nancy, Russ Housley, David Wagner, and Jesse Walker. "Security Flaws in 802.11 Data Link Protocols." Communications of the ACM, Vol 46, No 5 (May 2003): 35-39.

Housley, Russell. "Use of the RSAES-OAEP Key Transport Algorithm in Cryptographic Message Syntax (CMS)." RFC 3560, July 2003.

Whiting, Doug, Russell Housley, and Niels Ferguson. "Counter with CBC-MAC (CCM)." RFC 3610, September 2003.

Housley, Russell. "Using Advanced Encryption Standard (AES) Counter Mode With IPsec Encapsulating Security Payload (ESP)." RFC 3686, January 2004.

Housley, Russ. "Public Key Infrastructure (PKI)." *The Internet Encyclopedia*. New York: John Wiley & Sons, 2004.

Santesson, Stefan, Russell Housley, and Trevor Freeman. "Internet X.509 Public Key Infrastructure: Logotypes in X.509 Certificates." RFC 3709, February 2004.

Housley, Russell, and Tim Moore. "Certificate Extensions and Attributes Supporting Authentication in Point-to-Point Protocol (PPP) and Wireless Local Area Networks (WLAN)." RFC 3770, May 2004.

Arkko, Jari, and Russ Housley. "IETF Revises Extensible Authentication Protocol." Cipher Electronic Issue 60
(May 18, 2004). [http://www.ieee-security.org/Cipher/PastIssues/2004/E60.May-2004/E60.May-2004.txt]

Housley, Russell. "Cryptographic Message Syntax (CMS)." RFC 3852, July 2004.

Housley, Russ and Karen Seo. "IETF Updates IP Security Protocol (IPsec)." Cipher Electronic Issue 61 (July 17, 2004). [http://www.ieee-security.org/Cipher/PastIssues/2004/E61.Jul-2004/E61.Jul-2004.txt]

Housley, Russell. "A 224-bit One-way Hash Function: SHA-224." RFC 3874, September 2004.

Turner, Sean and Russ Housley. "IETF Revises Cryptographic Message Syntax and Secure Multipurpose Internet Mail Extensions." Cipher Electronic Issue 63 (November 18, 2004).
[http://www.ieee-security.org/Cipher/PastIssues/2004/E63.Nov-2004/E63.Nov-2004.txt]

Housley, Russ, Jesse Walker, and Nancy Cam-Winget. "Wireless Local Area Network Security: A Framework for Repairing the Broken WEP Protocol." In NTIA Special Publication SP-05-418, *Proceedings of the International Symposium on Advanced Radio Technologies*, March 2005, pp 81-89.

Hartman, Sam and Russ Housley. "IETF Revises Kerberos Specifications." Cipher Electronic Issue 65 (March 18, 2005). [http://www.ieee-security.org/Cipher/PastIssues/2005/E65.Mar-2005/E65.Mar-2005.txt]

Housley, Russell. "BinaryTime: An Alternate Format for Representing Date and Time in ASN.1." RFC 4049,
April 2005.

Housley, Russell. "Protecting Multiple Contents with the Cryptographic Message Syntax (CMS)." RFC 4073,
May 2005.

Lovick, Chris and Russ Housley. "Recent IETF Progress in Standardizing the SSH Protocol." Cipher Electronic Issue 66 (May 17, 2005). [http://www.ieee-security.org/Cipher/PastIssues/2005/E66.May-2005/E66.May-2005-2.txt]

Schaad, James, Burton Kaliski, and Russell Housley. "Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile." RFC 4055, June 2005.

Bellovin, Steven and Russell Housley. "Guidelines for Cryptographic Key Management." RFC 4107, BCP 107, June 2005.

Housley, Russell. "Using Cryptographic Message Syntax (CMS) to Protect Firmware Packages." RFC 4108, August 2005.

Housley, Russell. "Using Advanced Encryption Standard (AES) CCM Mode with IPsec Encapsulating Security Payload (ESP)." RFC 4309, November 2005.

Santesson, Stefan and Russell Housley. "Internet X.509 Public Key Infrastructure Authority Information Access Certificate Revocation List (CRL) Extension." RFC 4325, December 2005.

Housley, Russell. "Security Standards Activities", *Encyclopedia of Cryptography and Security*, H. van Tilborg (Ed.), Springer, 2005, pp. 552-558.

Housley, Russell, and Tim Moore. "Certificate Extensions and Attributes Supporting Authentication in Point-to-Point Protocol (PPP) and Wireless Local Area Networks (WLAN)." RFC 4334, February 2006.

Housley, Russell, and Stefan Santesson. "Update to DirectoryString Processing in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile." RFC 4630, August 2006.

Housley, Russell, and Alan Corry. "GigaBeam High-Speed Radio Link Encryption." RFC 4705, October 2006.

Housley, Russell. "Cryptographic Message Syntax (CMS) Multiple Signer Clarification." RFC 4853, April 2007.

Housley, Russell, and Bernard Aboba. "Guidance for Authentication, Authorization, and Accounting (AAA) Key Management." RFC 4962, BCP 132. July 2007.

Housley, Russell, and Jerome A. Solinas. "Suite B in Secure/Multipurpose Internet Mail Extensions (S/MIME)." RFC 5008, September 2007.

Housley, Russell. "Ask The Expert: Russ Housley." *DNSSEC Deployment Initiative*, 1 October 2007, pp 1. Interview.
[http://www.dnssec-deployment.org/news/dnssecthismonth/200710-dnssecthismonth/200710-dnssecthismonth.pdf]
[http://www.dnssec-deployment.org/news/dnssecthismonth/200710-dnssecthismonth/asktheexpert-housley.html]

Housley, Russell. "Cryptographic Message Syntax (CMS) Authenticated-Enveloped-Data Content Type."
RFC 5083, November 2007.

Housley, Russell. "Using AES-CCM and AES-GCM Authenticated Encryption in the Cryptographic Message Syntax (CMS)." RFC 5084, November 2007.

Freeman, Trevor, Russell Housley, Ambrish Malpani, David Cooper, and Server-Based Certificate Validation Protocol (SCVP)." RFC 5055, December 2007.

Cooper, David, Stefan Santesson, Stephen Farrell, Sharon Boeyen, Russell Housley, and William Polk. "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile." RFC 5280, May 2008.

Contreras, Jorge, and Russ Housley. "IP Squared: Internet Standards and Intellectual Property." *IEEE Internet Computing*, vol. 12, no. 6, pp. 83-86, November/December, 2008.

Turner, Sean, Daniel R. L. Brown, Kelvin Yiu, Russ Housley, and Tim Polk. "Elliptic Curve Cryptography Subject Public Key Information." RFC 5480, March 2009.

Housley, Russell. "Digital Signatures on Internet-Draft Documents." RFC 5485, March 2009.

Salter, Margaret, Eric Rescorla, and Russell Housley. "Suite B Profile for Transport Layer Security (TLS)."
RFC 5430, March 2009.

Housley, Russell, and Morris Dworkin. "Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm." RFC 5649, August 2009.

Housley, Russell. "Cryptographic Message Syntax (CMS)." RFC 5652, September 2009.

Alvestrand, Harald, and Russell Housley. "IESG Procedures for Handling of Independent and IRTF Stream Submissions." RFC 5742, December 2009.

Farrell, Stephen, Russell Housley, and Sean Turner. "An Internet Attribute Certificate Profile for Authorization." RFC 5755, January 2010.

Turner, Sean, Daniel Brown, Kelvin Yiu, Russell Housley, and William Polk. "Updates for RSAES-OAEP and RSASSA-PSS Algorithm Parameters." RFC 5756, January 2010.

Weiler, Samuel, David Ward, and Russell Housley. "The rsync URI Scheme." RFC 5781, February 2010.

Housley, Russell. " The application/pkix-attr-cert Media Type for Attribute Certificates." RFC 5877, May 2010.

Brown, Mark, and Russell Housley. "Transport Layer Security (TLS) Authorization Extensions." RFC 5878,
May 2010.

Housley, Russell, Sam Ashmore, and Carl Wallace. "Trust Anchor Format." RFC 5914, June 2010.

Housley, Russell, Sam Ashmore, and Carl Wallace. "Trust Anchor Management Protocol (TAMP)." RFC 5934, August 2010.

Turner, Sean, and Russell Housley. "Additional Cryptographic Message Syntax (CMS) Revocation Information Choices." RFC 5940, August 2010.

Housley, Russell. "BinaryTime: An Alternate Format for Representing Date and Time in ASN.1." RFC 6019,
September 2010.

Housley, Russell, Sam Ashmore, and Carl Wallace. "Cryptographic Message Syntax (CMS) Content Constraints Extension." RFC 6010, September 2010.

Turner, Sean, and Russell Housley. "Cryptographic Message Syntax (CMS) Symmetric Key Package Content Type." RFC 6031, December 2010.

Turner, Sean, and Russell Housley. "Cryptographic Message Syntax (CMS) Encrypted Key Package Content Type." RFC 6032, December 2010.

Santesson, Stefan, Russell Housley, Siddharth Bajaj, and Leonard Rosenthol. "Internet X.509 Public Key Infrastructure -- Certificate Image." RFC 6170, May 2011.

Housley, Russell, and Jerome A. Solinas. "Suite B in Secure/Multipurpose Internet Mail Extensions (S/MIME)." RFC 6318, June 2011.

Housley, Russell. "Conclusion of FYI RFC Sub-Series." RFC 6360, August 2011.

Housley, Russell, David Crocker, and Eric Burger. "Reducing the Standards Track to Two Maturity Levels."
RFC 6410, BCP 9, October 2011.

Salter, Margaret, and Russell Housley. "Suite B Profile for Transport Layer Security (TLS)." RFC 6460, January 2012.

Housley, Russell, Steve Mills, Jeff Jaffe, Bernard Aboba, and Lynn St.Amour. "Affirmation of the Modern Paradigm for Standards." RFC 6852, January 2013.

APPENDIX B – LIST OF MATERIALS CONSIDERED

1) U.S. Patent No. 6,502,135

2) U.S. Patent No. 7,490,151

3) U.S. Patent No. 7,418,504

4) U.S. Patent No. 7,921,211

5) Takahiro Kiuchi and Shigekoto Kaihara, "C-HTTP - The Development of a Secure, Closed HTTP-based Network on the Internet," published in the Proceedings of SNDSS 1996

6) C. I. Dalton and J. F. Griffin, "Applying Military Grade Security to the Internet," published in the Proceedings of JENC8, May 1997

7) RFC 1123

8) VirnetX, Inc. v. Microsoft Corporation – Memorandum Opinion

9) W. Richard Stevens, "TCP/IP Illustrated, Volume 1: The Protocols" (Reading, Mass.: Addison-Wesley, 1994), page 187

10) Windows Sockets – An Open Interface for Network Programming Under Microsoft Windows, Version 1.1, January 20, 1993 (copy obtained from http://www.sockets.com/winsock.htm)

11) eCOS Reference Manual, Chapter 38, TCP/IP Library Reference, gethostbyname, March 13, 1997 (copy obtained from http://www.ecos.sourceware.org/docs-2.0/ref/net-common-tcpip-manpages-gethostbyname.html)

12) Definitions of HTML and HTTP from Computer Desktop Encyclopedia, 9th Edition, The McGraw-Hill Companies, 2001, pp. 435-436

13) U.S. Patent No. 6,560,634

14) File History of Application No. 10/714,849

15) File History of Application No. 11/840,560

16) Cryptography and Network Security: *Principles and Practice*, Second Edition, William Stallings, Chapter 13

17) RFC 1738

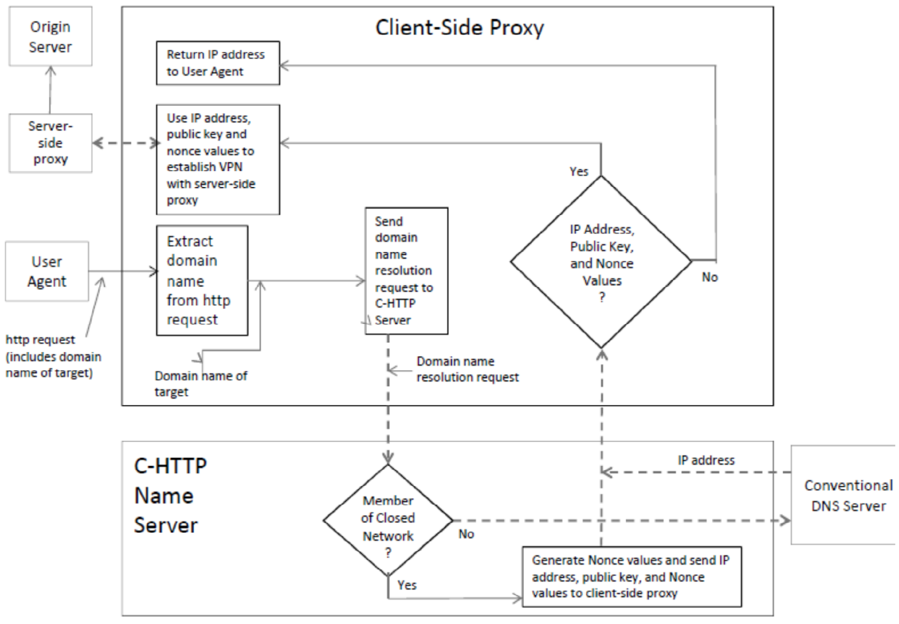
18) RFC 1034

19) U.S. Patent No. 6,449,657 (Stanbach, Jr. et al.)

20) U.S. Patent No. 6,546,003 (Farris)

APPENDIX C – KIUCHI FOR REQUEST 1

U.S. Patent No. 7,490,151	Claims 1, 13 are unpatentable under 35 U.S.C. § 103 as being obvious in view of Kiuchi
	Overview: The ‘151 patent makes clear that software modules such as a DNS proxy server module and a DNS module can reside on separate machines or be combined in ways where various functions reside on the same machine. (Ex.1001 at 38:30-35) For example, the ‘151 Patent makes clear that the DNS proxy and DNS server can be combined into a single server or can operate in separate servers. (Ex.1001 at 38:30-33). This is the nature of software, where developers generally have great leeway in how to divide functions into software modules and where to place the software modules. Kiuchi also teaches to an ordinarily skilled artisan that the functions of a “DNS proxy server” can be included in the same machine as the client computer (i.e., the client-side proxy machine) or in a DNS server such as the C-HTTP name server, and that the functions of a “gatekeeper computer” can be included in the server-side proxy or the C-HTTP server. Kiuchi clearly recognizes and discloses that a conventional DNS lookup for the domain name is needed when access is not being requested to a secure server, i.e., when the requested server-side proxy is not registered in the closed network. (Ex.1002 at 65, sec. 2.3(2)). This is identical to the ‘135 Patent, where a DNS lookup is performed when access is not being requested to a secure server. (ex.1001 at 38:12-16). Kiuchi defines three new components for the system, namely the client-side proxy, the server-side proxy, and the C-HTTP name server. (Ex.1002 at 64, sec. 2.1). While Kiuchi describes a system in which a conventional DNS lookup request is made from the client-side proxy, it would have been apparent to a person of ordinary skill in the art

	based on Kiuchi's teachings to make the conventional DNS lookup request from the C-HTTP name server. As discussed above, the C-HTTP name server already determines whether the DNS request received from the client-side proxy is requesting access to a secure server (i.e., a server-side proxy). Rather than returning an error status to the client-side proxy when the DNS request is not requesting access to a secure server, it would have been trivial and obvious as a mere design choice for the C-HTTP name server to pass the domain name received in the C-HTTP name service request to the conventional DNS server (i.e., a "DNS function"), as depicted in the following Figure:  Origin Server Server-side proxy User Agent http request (includes domain name of target) Client-Side Proxy Return IP address to User Agent Use IP address, public key and nonce values to establish VPN with server-side proxy Extract domain name from http request Domain name of target Send domain name resolution request to C-HTTP Server Domain name resolution request IP Address, Public Key, and Nonce Values ? Yes No C-HTTP Name Server Member of Closed Network ? No Yes Generate Nonce values and send IP address, public key, and Nonce values to client-side proxy IP address Conventional DNS Server Figure 3 – Alternate Functional Diagram of Kiuchi Such a configuration, which places a DNS proxy server function in a modified C-HTTP name server (similar to placement of the DNS proxy server function of the '135 patent in the DNS server – see Ex.1001 at FIG. 26), is merely a rearrangement of existing functions within the C-HTTP system and could be implemented with or without modifying Kiuchi's protocols. For example, a C-HTTP name service response message containing an IP address without a public key and Nonce values (e.g., using values of

	zero or other convention for the public key and Nonce fields, or modifying the protocol to use a previously unused flag in the response to indicate that a public key and Nonce values are not provided) would indicate to the client-side proxy that the DNS request is not requesting access to a secure server and hence that no secure/encrypted channel is needed. The motivation for modifying Kiuchi in this way would have been to streamline the operation of the system, e.g., instead of having the C-HTTP name server send an error status to the client-proxy which would in turn initiate a conventional DNS inquiry, the modification eliminates the error status message from the process by having the C-HTTP name server directly initiate the request to the conventional DNS server. Thus, the combination of claim 8 is obvious in view of Kiuchi. See *KSR Intern. Co. v. Teleflex Inc.*, 550 U.S. 398, 401; 127 S.Ct. 1727, 173 (2007) ("a combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results"); *Sakraida v. Ag Pro, Inc.*, 425 U.S. 273, 282, 96 S.Ct. 1532 (1976) (when a patent "simply arranges old elements with each performing the same function it had been known to perform" and yields no more than one would expect from such an arrangement, the combination is obvious). For purposes of this analysis, the modified C-HTTP name server includes a "DNS proxy server module" or "DNS module," the client-side proxy is the "client," and the server-side proxy is the "secure computer." In response to receiving an HTTP request from the user agent, the client-side proxy sends a DNS request to the C-HTTP name server. The DNS request, which contains the domain name from the HTTP request (i.e., the hostname given to the server-side proxy), requests the IP address of the server-side proxy (i.e., the "secure server"). The C-HTTP is a data processing device having a memory storing a "DNS proxy module" or "DNS module" that intercepts DNS requests sent by the client-side proxy (i.e., the "client") and performs the steps

	recited in the claims. The client-side proxy is a "client" because it is a computer that sends a DNS request. The server-side proxy is a "secure server" because it is a server that requires authorization for access and can communicate in an encrypted channel. A secure/encrypted channel is automatically initiated/created between the client-side proxy and the server-side proxy if the DNS request corresponds to a secure server (i.e., a server-side proxy).
[1.0a] A data processing device, comprising memory [1.0b] storing a domain name server (DNS) proxy module [1.0c] that intercepts DNS requests sent by a client and, [1.0d] for each intercepted DNS request, performs the steps of:	[l.0a-d] *A data processing device, comprising memory storing a domain name server (DNS) proxy module that intercepts DNS requests sent by a client and, for each intercepted DNS request, performs the steps of:* The modified C-HTTP name server including a DNS proxy server module is a data processing device. The functions performed in the C-HTTP name server are necessarily stored in computer program code in memory, as is the case for any such processing device. The modified C-HTTP name server including a DNS proxy server module intercepts DNS requests sent by the client-side proxy (i.e., the "client"). For each intercepted DNS request, the DNS proxy server module in the modified C-HTTP name server performs the steps recited in the claims
[1.1] (i) determining whether the intercepted DNS request corresponds to a secure server;	*[1.1] determining whether the intercepted DNS request corresponds to a secure server:* The C-HTTP name service request is a "DNS request" because it is a communication that contains a domain name (i.e., the hostname from the URL in the HTTP request) and requests an IP address for the domain name. In particular, Kiuchi teaches that the client-side proxy generates a Domain Name Service request: "A client-side proxy asks the C-HTTP name server whether it can communicate with the host specified in a given URL."

	(Kiuchi, at 65). In a manner similar to that described at columns 37-38 of the ‘151 Patent, the C-HTTP name server may perform the “determining” step. Upon receiving the C-HTTP name service request (i.e., DNS request) from the client-side proxy, the DNS proxy module of the modified C-HTTP name server examines whether the requested server-side proxy is registered in the closed network. In particular: A client-side proxy asks the C-HTTP name server whether it can communicate with the host specified in a given URL. If the name server confirms that the query is legitimate, it *examines whether the requested server-side proxy is registered in the closed network* and is permitted to accept the connection from the client-side proxy. (Ex. 1002 at 65, sec. 2.3(2), emphasis added) The DNS proxy module determines whether the intercepted DNS request sent by the client-side proxy corresponds to a server-side proxy in the closed network (i.e., a “secure server”) based on whether the requested server-side proxy is registered in the closed network. The DNS proxy server module determines that the DNS request corresponds to a secure server, if the requested server-side proxy is registered in the closed network. The DNS proxy server module determines that the DNS request does not correspond to a secure server, if the requested server-side proxy is not registered in the closed network. (Ex.1002 at 65, sec. 2.3(2)-(3)). Thus, step (1) of claim 1 is satisfied by the determination made by the DNS proxy server module of the modified C-HTTP name server. Note that the ‘151 Patent provides, as an example of “determining,” making “reference to an internal table of such sites.” (see ‘151 Patent at column 37, lines 64-65). In Kiuchi, the C-HTTP name server makes reference to an

	internal registry (i.e., "examines whether the requested server-side proxy is registered") to determine whether the DNS request is requesting access to a secure web site. (Kiuchi at 65, sec. 2.2).
[1.2a] (ii) when the intercepted DNS request does not correspond to a secure server, forwarding the DNS request to a DNS function [1.2b] that returns an IP address of a nonsecure computer, and	[1.2a-b] *when the intercepted DNS request does not correspond to a secure server, forwarding the DNS request to a DNS function that returns an IP address of a non secure computer:* As discussed above for step (1), the DNS proxy server module determines that the DNS request does not correspond to a secure server, if the requested server-side proxy is not registered in the closed network. When the DNS proxy server module makes this determination, it performs a DNS lookup to the conventional DNS server (i.e., a DNS function that returns an IP address of a nonsecure computer), which involves forwarding the domain name (DNS request) to the conventional DNS server and receiving an IP address back from the DNS server. Specifically, Kiuchi teaches that "If a client-side proxy receives an error status, then it performs DNS lookup, behaving like an ordinary HTTP/1.0 proxy." (Ex.1002 at 65, section 2.3, paragraph 2). In the modified C-HTTP server, this DNS lookup function resides in the C-HTTP name server rather than in the client-side proxy. Thus, the modified C-HTTP server including the "DNS proxy server module" or "DNS module" performs a DNS lookup if the requested server-side proxy is not registered in the closed network and hence performs a DNS lookup when the DNS request does not correspond to a secure server. A person of ordinary skill would have understood that Kiuchi's reference to performing a "DNS lookup ... like an ordinary HTTP/1.0 proxy" involves forwarding the DNS request to the conventional DNS server and receiving an IP address from the DNS server. For example, RFC 1123 (Ex.1010) defines how computers on the Internet should operate and states that when using

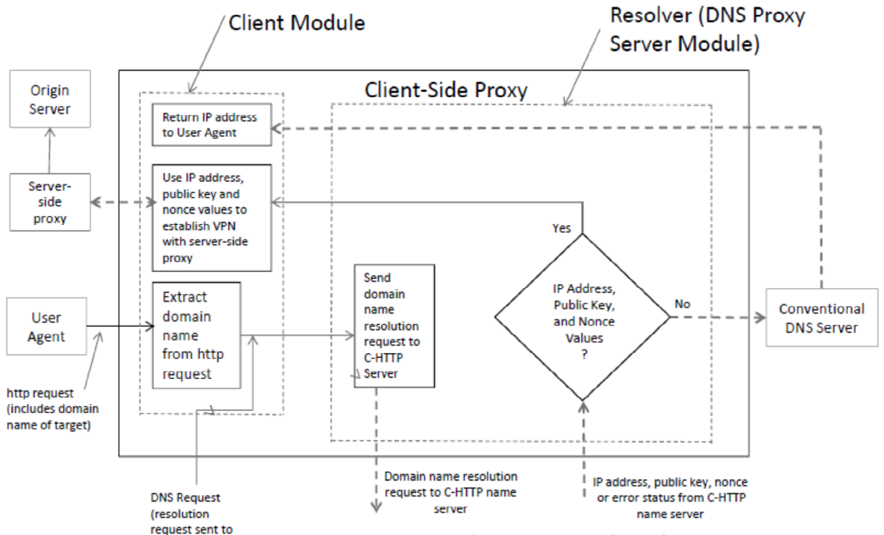
	domain names, *"Host domain names MUST be translated to IP addresses* as described in Section 6.1." (Ex.1010 at 13 (emphasis added).) Section 6.1, in turn, states that *"Every host MUST implement a resolver for the Domain Name System* (DNS), and it MUST implement a mechanism using this DNS resolver to convert host names to IP addresses and vice-versa." *(ld.* at 72.). Thus, in response to determining that the DNS request does not correspond to a secure server, the modified C-HTTP name server with "DNS proxy server module" or "DNS module" forwards the DNS request to the conventional DNS server.
[1.3a] (iii) when the intercepted DNS request corresponds to a secure server, [1.3b] automatically initiating an encrypted channel between the client and the secure server.	[1.3a] *when the intercepted DNS request corresponds to a secure server, automatically initiating an encrypted channel between the client and the secure server:* As discussed above for step (1), the DNS proxy server module determines that the DNS request corresponds to a secure server, if the requested server-side proxy is registered in the closed network. When the DNS proxy server module makes this determination, it sends a C-HTTP name service response to the client-side proxy containing the IP address and public key of the server-side proxy (i.e., the "secure server") as well as request and response Nonce values. In turn, the client-side proxy "sends a request for connection to the server-side proxy," which is encrypted using the server-side proxy's public key and contains the client-side proxy's IP address, hostname, request Nonce value and symmetric data exchange key for request encryption." (Ex.1002 at 65, right column, section 2.3(3)). The sending of the C-HTTP name service response by the DNS proxy server module to the client-side proxy constitutes "automatically initiating" a secure, encrypted channel within the context of the claim because the C-HTTP name service response causes the client-side proxy to send a request for connection to the server-side proxy. (Ex.1002 at 65, sec. 2.3(3)). In this regard, it should be noted that the '151 Patent provides, as one example of automatically initiating/creating a secure, encrypted channel, transmission of a message

	requesting that a VPN be created (see Ex.1001 at 37:66-38:2). The C-HTTP name service response is analogous because it is a message that causes the secure, encrypted channel to be created. Kiuchi teaches that the C-HTTP name server automatically initiates an encrypted channel (e.g., a VPN) between the client and the secure server when the intercepted DNS request corresponds to a secure server. Furthermore, as discussed in portion [1.0a], Kiuchi teaches that all communications between the client-side proxy and the server-side proxy are encrypted: Once the connection is established, a client-side proxy forwards HTTP/1.0 requests from the user agent *in encrypted form* using C-HTTP format. (Kiuchi at 66.) Accordingly, Kiuchi teaches each portion of [1.3a-b]. Therefore, step (3) of claim 1 is satisfied by sending of the C-HTTP name service response message to the client-side proxy in response to determining that the DNS request corresponds to a secure server. Thus, Kiuchi teaches all the steps in the method claim 1.
[13.0a] A computer readable medium [13.0b] storing a domain name server (DNS) module [13.0c] comprised of computer readable	[13.0a-c] *A computer readable medium storing a domain name server (DNS) module comprised of computer readable instructions that, when executed, cause a data processing device to perform the steps of:* Kiuchi describes the C-HTTP name server, which is a data processing device. The C-HTTP name server necessarily includes a computer readable medium with executable, computer readable instructions.

instructions that, when executed, cause a data processing device to perform the steps of:	
[13.1] (i) determining whether a DNS request sent by a client corresponds to a secure server;	[13.1] *determining whether a DNS request sent by a client corresponds to a secure server:* See claim portion [1.1], which is identical except for claim 13 reciting a "DNS request sent by a client" instead of "intercepted DNS request." As discussed above, Kiuchi teaches that the client-side proxy sends a Domain Name Service request: "A client-side proxy asks the C-HTTP name server whether it can communicate with the host specified in a given URL." (Kiuchi, at 65).
[13.2a] (ii) when the DNS request does not correspond to a secure server, forwarding the DNS request to a DNS function [13.2b] that returns an IP address of a non secure computer; and	[13.2a]-[13.2b] *(ii) when the DNS request does not correspond to a secure server, forwarding the DNS request to a DNS function that returns an IP address of a nonsecure computer:* See claim portions [1.2a-1.2b], which are identical.
[13.3a] (iii)	[13.3a] *when the intercepted DNS request corresponds to a*

when the intercepted DNS request corresponds to a secure server, [13.3b] automatically creating a secure channel between the client and the secure server.	*secure server:* See claim portion [1.3a], which is identical. [13.3b] *automatically creating a secure channel between the client and the secure server:* See claim portion [1.3b], which is identical except that claim 13 recites a "secure channel" instead of "an encrypted channel." Kiuchi states that the client-side proxy and the server-side proxy "communicate with each other using a secure, encrypted protocol." (Kiuchi at 64, Abstract) Further, the use of public key and Nonce values create an encrypted channel which is therefore secure since those VPN resources are only made available to the client and the target server. There is no user involvement in the creation of the secure channel. It is automatic. Just as the '151 patent describes "Use of a DNS Proxy to Transparently Create Virtual Private Networks" by sending a message to a gatekeeper, so too does Kiuchi disclose creating a virtual private network when the C-HTTP name server sends out public key and Nonce values in response to a DNS request.

APPENDIX D – KIUCHI FOR REQUEST 2

U.S. Patent No. 7,490,151	Claims 1, 13 are unpatentable under 35 U.S.C. § 102 as anticipated by Kiuchi
	<u>Overview:</u> A careful consideration of the inner workings of the client-side proxy reveals that Kiuchi's client-side proxy performs a "resolver" function that receives a domain name resolution request from an internal client (in this case, the domain name extracted from the received HTTP request) and returns an IP address for the domain name. The following figure schematically shows the resolver and client functions in the client-side proxy:  Client Module Resolver (DNS Proxy Server Module) Client-Side Proxy Origin Server Server-side proxy User Agent http request (includes domain name of target) Return IP address to User Agent Use IP address, public key and nonce values to establish VPN with server-side proxy Extract domain name from http request Send domain name resolution request to C-HTTP Server Yes IP Address, Public Key, and Nonce Values ? No Conventional DNS Server DNS Request (resolution request sent to Resolver) Domain name resolution request to C-HTTP name server IP address, public key, nonce or error status from C-HTTP name server Figure 4 - Implementation of Kiuchi Resolver functions were known in the art well before the '151 Patent was filed. Thus, in Kiuchi, an internal resolution request (which is a "DNS request" because it is a communication that contains a domain name and requests an IP address for the domain name) is made to the resolver function (which is a collection of software functions within

	the client-side proxy), which acts as a DNS proxy server to contact the C-HTTP name server and optionally also the conventional DNS server to obtain an IP address for the domain name and return the IP address to the internal client. Furthermore, a careful consideration of the inner workings of the client-side proxy also reveals that the resolver function performs functions that map directly to the functions performed by the DNS proxy server of the '151 Patent. The DNS proxy server of the '151 Patent receives a DNS request, determines whether access to a secure web site has been requested (e.g., based on a domain name extension or by reference to an internal table of such sites), automatically initiates a VPN if access to a secure target web site has been requested, and passes through the DNS request to a conventional DNS server if access to a non-secure site had been requested (see Ex.1001 at 37:60-38:11). Similarly, the resolver function of Kiuchi receives a DNS request, determines whether access to a secure web site has been requested (based on a query to the C-HTTP name server, which essentially is just a remote table lookup similar to the internal table lookup of the '151 Patent), automatically initiates a VPN if access to a secure target web site has been requested, and passes through the DNS request to a conventional DNS server if access to a non-secure site had been requested. Thus, Kiuchi's resolver function is a DNS proxy server because it is a computer or program that responds to a domain name inquiry in place of a DNS. Consequently, Kiuchi's client-side proxy effectively includes a Client Module and a DNS Proxy Server Module, as shown in the figure above. For purposes of the following analysis, the Client Module of the client-side proxy is the "client," the server-side proxy is the "secure server," and the DNS Proxy Server Module is a "DNS proxy server module" in the client-side proxy. In response to receiving an HTTP request from the user

	agent, the Client Module sends the domain name from the URL (i.e., a "DNS request") to the DNS Proxy Server Module. This domain name will be the hostname of the server-side proxy, if the HTTP request is directed to a resource on the origin server. The client-side proxy is a data processing device having a memory storing a "DNS proxy module" or "DNS module" that intercepts DNS requests sent by the Client Module (i.e., the "client") and performs the steps recited in the claims. The Client Module is a "client" because it is a program that sends a DNS request. The server-side proxy is a "secure server" because it is a server that requires authorization for access and can communicate in an encrypted channel. A secure, encrypted channel is automatically initiated/created between the client-side proxy and the server-side proxy if the DNS request corresponds to the server-side proxy (i.e., the "secure server").
[1.0a] A data processing device, comprising memory [1.0b] storing a domain name server (DNS) proxy module [1.0c] that intercepts DNS requests sent by a client and, [1.0d] for each intercepted DNS request, performs the steps of:	[l.0a-d] *A data processing device, comprising memory storing a domain name server (DNS) proxy module that intercepts DNS requests sent by a client and, for each intercepted DNS request, performs the steps of:* The client-side proxy including the DNS Proxy Server Module is a data processing device. The functions performed in the client-side proxy, including the DNS Proxy Server Module, are necessarily stored in computer program code in memory, as is the case for any such processing device.. The DNS Proxy Server Module intercepts DNS requests sent by the Client Module (i.e., the "client"). For each intercepted DNS request, the DNS Proxy Server Module in the modified C-HTTP name server performs the steps recited in the claims.
[1.1] (i) determining whether the	*[1.1] determining whether the intercepted DNS request corresponds to a secure server:*

intercepted DNS request corresponds to a secure server;	When the Client Module (i.e., the "client") receives the HTTP request from the user agent, it extracts the hostname (i.e., domain name) from the URL received in the HTTP request and sends an internal resolver request containing the domain name to the DNS Proxy Server Module. This internal resolver request is a "DNS request" because it is a communication that contains a domain name (i.e., the hostname from the URL in the HTTP request) and requests an IP address for the domain name. If the domain name sent in the DNS request is the hostname given to the server-side proxy, then the DNS request corresponds to a secure server (i.e., the server-side proxy). Upon receiving the DNS request from the Client Module, the DNS Proxy Server Module sends the domain name to the C-HTTP name server in the form of a C-HTTP name service request to ask the C-HTTP name server whether the client-side proxy can communicate with the specified host. (Ex.1002 at 65, sec. 2.3(2)). The C-HTTP name server "examines whether the requested server-side proxy is registered in the closed network." (Ex.1002 at 65, sec. 2.3(2)). The C-HTTP name server sends a C-HTTP name service response to the DNS Proxy Server Module containing "the IP address and public key of the server-side proxy and both request and response Nonce values" (Ex.1002 at 65, sec. 2.3(2)), if the requested server-side proxy is registered in the closed network. The C-HTTP name server sends "a status code which indicates an error," if the requested server-side proxy is not registered in the closed network. (Ex.1002 at 65, sec. 2.3(2)-(3)). The DNS Proxy Server Module determines whether the DNS request sent by the Client Module corresponds to a secure server based on the type of response received from the C-HTTP name server. In particular, the DNS Proxy Server Module determines that the DNS request corresponds to a secure server, only if a C-HTTP name service response is returned, and determines that the DNS request does not correspond to a secure server, if the response is an error status.

	Thus, step (1) of claim 1 is satisfied by the determination made by the DNS Proxy Server Module based on the type of response received from the C-HTTP name server.
[1.2a] (ii) when the intercepted DNS request does not correspond to a secure server, forwarding the DNS request to a DNS function [1.2b] that returns an IP address of a nonsecure computer, and	[1.2a-b] *when the intercepted DNS request does not correspond to a secure server, forwarding the DNS request to a DNS function that returns an IP address of a non secure computer:* As discussed above for step (1), the DNS Proxy Server Module determines that the DNS request does not correspond to a secure server, if the response from the C-HTTP name server is an error status. When the DNS Proxy Server Module makes this determination, it performs a DNS lookup to the conventional DNS server (i.e., a DNS function that returns an IP address of a nonsecure computer), which involves forwarding the domain name (DNS request) to the conventional DNS server and receiving an IP address back from the DNS server. Specifically, Kiuchi teaches that "If a client-side proxy receives an error status, then it performs DNS lookup, behaving like an ordinary HTTP/1.0 proxy." (Ex.1002 at 65, section 2.3, paragraph 2). Thus, the client-side proxy, and more specifically the DNS Proxy Server Module in the client-side proxy, performs a DNS lookup in response to receiving the error status and hence performs a DNS lookup when the DNS request does not correspond to a secure server. A person of ordinary skill would have understood that Kiuchi's reference to performing a "DNS lookup ... like an ordinary HTTP/1.0 proxy" involves forwarding the DNS request to the conventional DNS server and receiving an IP address from the DNS server. For example, RFC 1123 (Ex.1010) defines how computers on the Internet should operate and states that when using domain names, *"Host domain names MUST be translated to IP addresses* as described in Section 6.1." (Ex.1010 at 13 (emphasis added).)

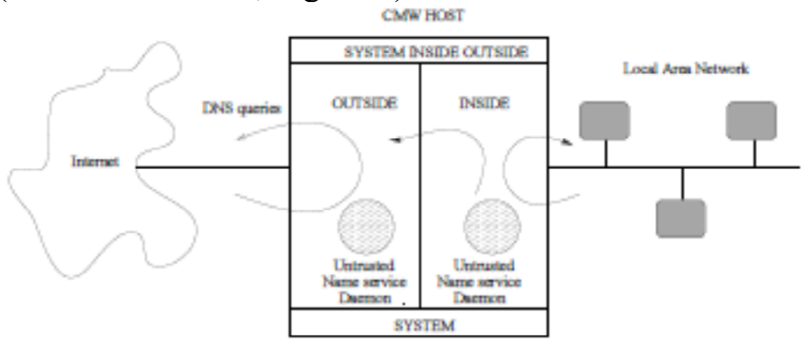
	Section 6.1, in turn, states that *"Every host MUST implement a resolver for the Domain Name System* (DNS), and it MUST implement a mechanism using this DNS resolver to convert host names to IP addresses and vice-versa." *(ld.* at 72.). Thus, in response to determining that the DNS request does not correspond to a secure server, the DNS Proxy Server Module forwards the DNS request to the conventional DNS server.
[1.3a] (iii) when the intercepted DNS request corresponds to a secure server, [1.3b] automatically initiating an encrypted channel between the client and the secure server.	[1.3a] *when the intercepted DNS request corresponds to a secure server, automatically initiating an encrypted channel between the client and the secure server:* Kiuchi teaches that the DNS Proxy Server Module automatically initiates an encrypted channel (e.g., a VPN) between the client and the secure server when the intercepted DNS request corresponds to a secure server. As discussed above for step (1), the DNS Proxy Server Module determines that the DNS request corresponds to a secure server, only if the response from the C-HTTP name server is a C-HTTP name service response. When the DNS Proxy Server Module makes this determination, it sends the IP address and VPN resources (e.g., the public key and Nonce values) received in the C-HTTP name service response to the Client Module. In turn, the Client Module "sends a request for connection to the server-side proxy," which is encrypted using the server-side proxy's public key and contains the client-side proxy's IP address, hostname, request Nonce value and symmetric data exchange key for request encryption." (Ex.1002 at 65, right column, section 2.3(3)). The sending of the IP address and VPN resources by the DNS Proxy Server Module to the Client Module constitutes "automatically initiating" a secure, encrypted channel within the context of the claim because this transaction causes the client-side proxy to send a request for connection to the

	server-side proxy. (Ex.1002 at 65, sec. 2.3(3)). In this regard, it should be noted that the '151 Patent provides, as one example of automatically initiating/creating a secure, encrypted channel, transmission of a message requesting that a VPN be created (see Ex.1001 at 37:66-38:2). Sending the IP address and VPN resources by the DNS Proxy Server Module to the Client Module is analogous because it is a message that causes the secure, encrypted channel to be created. Accordingly, Kiuchi teaches "automatically initiating the encrypted channel between the client and the secure server." In addition, Kiuchi teaches that the channel is encrypted: When the server-side proxy obtains the client-side proxy's IP address, hostname and public key, it authenticates the client-side proxy, checks the integrity of the request and the request Nonce value and generates both a connection ID derived from the server-side proxy's name, date and random numbers (32 bits) using MD5, and also *a second symmetric data exchange key for response encryption, which are sent to the client-side proxy.* When the client-side proxy accepts and checks them, *the connection is established.* (Kiuchi at 66, emphasis added). Further, Kiuchi teaches that all communications between the client-side proxy and the server-side proxy are encrypted: Once the connection is established, a client-side proxy forwards HTTP/1.0 requests from the user agent *in encrypted form* using C-HTTP format. (Kiuchi at 66.) Thus, Kiuchi teaches all the steps in the method and anticipates each portion of claim 1.
[13.0a] A computer readable	[13.0a-c] *A computer readable medium storing a domain name server (DNS) module comprised of computer readable instructions that, when executed, cause a data processing*

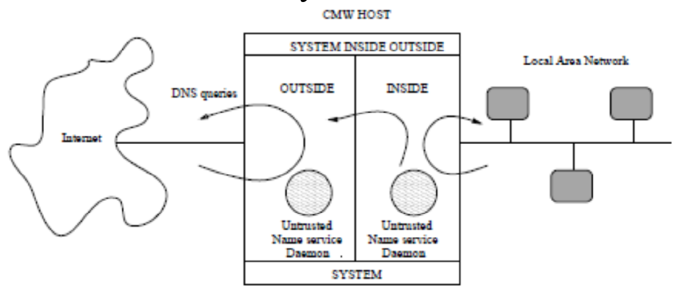
medium [13.0b] storing a domain name server (DNS) module [13.0c] comprised of computer readable instructions that, when executed, cause a data processing device to perform the steps of:	*device to perform the steps of:* As discussed in claim portion [1.0], the client-side proxy including the DNS Proxy Server Module is a data processing device. The functions performed in the client-side proxy, including the DNS Proxy Server Module (a DNS module), are necessarily stored in computer program code in memory, as is the case for any such processing device. The DNS Proxy Server Module intercepts DNS requests sent by the Client Module (i.e., the "client"). For each intercepted DNS request, the DNS Proxy Server Module in the modified C-HTTP name server performs the steps recited in the claims.
[13.1] (i) determining whether a DNS request sent by a client corresponds to a secure server;	[13.1] *determining whether a DNS request sent by a client corresponds to a secure server:* See claim portion [1.1], which is identical except for claim 13 reciting a "DNS request sent by a client" instead of "intercepted DNS request."
[13.2a] (ii) when the DNS request does not correspond to a secure server, forwarding the DNS request to a DNS function [13.2b] that returns an IP	[13.2a]-[13.2b] *(ii) when the DNS request does not correspond to a secure server, forwarding the DNS request to a DNS function that returns an IP address of a nonsecure computer:* See claim portions [1.2a-1.2b], which are identical.

address of a nonsecure computer; and	
[13.3a] (iii) when the intercepted DNS request corresponds to a secure server, [13.3b] automatically creating a secure channel between the client and the secure server.	[13.3a] *when the intercepted DNS request corresponds to a secure server:* See claim portion [1.3a], which is identical. [13.3b] *automatically creating a secure channel between the client and the secure server:* See claim portion [1.3b], which is identical except that claim 13 recites a "secure channel" instead of "an encrypted channel." Kiuchi states that the client-side proxy and the server-side proxy "communicate with each other using a secure, encrypted protocol." (Kiuchi at 64, Abstract) Further, the use of public key and Nonce values create an encrypted channel which is therefore secure since those VPN resources are only made available to the client and the target server. There is no user involvement in the creation of the secure channel. It is automatic. Just as the '151 patent describes "Use of a DNS Proxy to Transparently Create Virtual Private Networks" by sending a message to a gatekeeper, so too does Kiuchi disclose creating a virtual private network when the C-HTTP name server sends out public key and Nonce values in response to a DNS request.

APPENDIX E – DALTON/KIUCHI

U.S. Patent No. 7,490,151*	Claims 1, 13 are unpatentable under 35 U.S.C. § 103 as being obvious over Dalton in view of Kiuchi
	Overview: Dalton described a firewalled Domain Name System in a single machine, referred to as a Compartmented Mode Workstation (CMW) that provides a DNS service such that clients on an internal (closed) network can have access to both public and private hosts on the internal network as well as access to hosts on an external network, while hosts on the external network can have access through the CMW to only public hosts on the internal network. Dalton illustrates a simple example of the CMW-based system having two zones (i.e., internal/protected and external/public) as follows (Ex.1003 at 711-4, Figure 2):  CMW HOST SYSTEM INSIDE OUTSIDE OUTSIDE INSIDE Local Area Network DNS queries Internet Untrusted Name service Daemon Untrusted Name service Daemon SYSTEM The CMW includes a front end daemon that intercepts a DNS request from an internal client. "[T]he front end daemon would be configured … to proxy packets at the request of the SYSTEM INSIDE name service daemon over either the SYSTEM INSIDE or SYSTEM OUTSIDE networks. This allows the internal name server to query other non-local name servers out on the Internet in order to resolve an external address for an internal client …." (Ex.1003 at 711-4) Kiuchi addressed the hospital space, recognizing that its technology was equally applicable for use by other institutions. Kiuchi explicitly explained the incentive to

	privately communicate over the Internet: The Internet is expected to become available to almost all major hospitals. Although a closed network can be constructed using a privately-leased circuit, additional investment for its construction is necessary. If a closed network can be constructed on the Internet, it would be *convenient, speedy and reasonable in terms of cost*. In addition, if a closed network is realized by privately leased circuits, it is not always easy to operate several closed networks flexibly and simultaneously. (Ex. 1002, p. 69, sec. 4.5, emphasis added) Kiuchi disclosed "a closed HTTP-based virtual network [that] can be constructed for closed groups; for example, the headquarters and branches of a given corporation." (Ex. 1002, p. 69, sec. 5) Kiuchi re-emphasized the cost incentive for moving to the Internet: "This kind of usage may not fit with the spirit of the Internet, but if resources which might otherwise be invested in private circuits are channeled into the Internet, it will contribute to its further development." *Id.* The Local Area Network of Dalton would have been a costly solution for some institutions at the time. As Kiuchi explained the convenience, speed and costs associated with a private network implementation on the Internet were a great incentive for institutions. Thus, those of ordinary skill in the art would have been clearly motivated to replace Dalton's closed, private local area network with a closed HTTP-based virtual private network.
[1.0] A data processing device, comprising	*A data processing device, comprising memory storing a domain name server (DNS) proxy module that intercepts DNS requests sent by a client and, for each intercepted DNS request, performs the steps of:*

memory storing a domain name server (DNS) proxy module that intercepts DNS requests sent by a client and, for each intercepted DNS request, performs the steps of:	Dalton's CMW is a data processing device. The functions performed in the CMW are necessarily stored as computer program code in memory, as is the case for any such processing device.
[1.1] (i) determining whether the intercepted DNS request corresponds to a secure server;	*[1.1] determining whether the intercepted DNS request corresponds to a secure server:* The DNS request received by the CMW is a "DNS request." In particular, Dalton implements a conventional DNS, in which DNS queries request an IP address associated with a domain name. Dalton discloses that "DNS provides a mapping between host names and numerical Internet Protocol addresses". (Ex.1003 at 711-3, first column, third paragraph). Dalton makes clear that the Domain Name System (DNS) used in the CMW-based DNS service is the conventional DNS defined by the IETF. (see Ex.1003 at 711-4, first column, third paragraph, referencing IETF Request For Comments (RFC) 1034 and 1035 – Dalton references [5,6]). Thus, the requests are DNS requests. Dalton illustrates its system as follows:  CMW HOST SYSTEM INSIDE OUTSIDE Local Area Network DNS queries OUTSIDE INSIDE Internet Untrusted Name service Daemon Untrusted Name service Daemon SYSTEM

	(Ex. 1003, Figure 2) Dalton describes how the DNS request from an internal client (i.e., the client computer) on the local area network does not go directly to a DNS function, but is first intercepted by the front end daemon of Dalton's CMW: "The main body of the front end daemon (MLNAMED) is implemented as a continuous loop. *It sits and waits for incoming DNS queries* (both TCP and UDP) and for answers sent back from the untrusted name service daemons that require forwarding over the network to the original querying clients." (Ex.1003 at 711-4, first column, third paragraph, emphasis added). The CMW performs the "determining whether the intercepted DNS request corresponds to a secure server" based on the presence or absence of a DNS record for the domain name in the DNS request. The DNS data held by the name service daemon running at the SYSTEM INSIDE level forms the basis for determining whether an internal DNS client is requesting access to one of the hosts on the closed internal network: …the name service daemon running at the SYSTEM INSIDE level holds a full set of DNS data for that zone. Internal querying clients therefore have access to all the DNS records for their zone (Dalton at 711-3, second column, fourth paragraph). Thus, this name service daemon is responsible for resolving queries received from internal DNS clients. Further, Dalton makes clear that a host on the Local Area Network (LAN) can be a World Wide Web server, describing "locally provided public services such as World Wide Web servers." (Ex.1003 at 711-3, second column, fourth paragraph) Thus, the WWW server is just one example of a server on the LAN.

Dalton illustrates its system in more depth as follows:

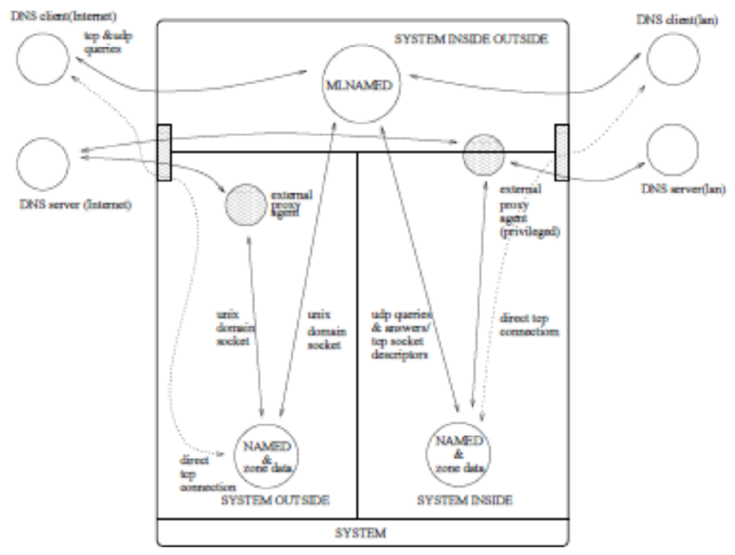


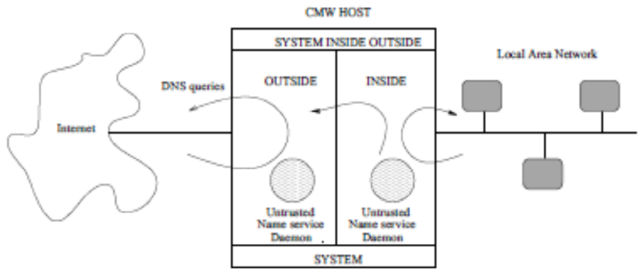
Figure 3: Detailed MLNAMED / NAMED architecture

(Ex. 1003, Figure 3)

Figure 3 demonstrates that the front end daemon (MLNAMED) is configured to proxy packets at the request of the SYSTEM INSIDE name service daemon over either the SYSTEM INSIDE or SYSTEM OUTSIDE networks. Further, the Figure teaches that the front end daemon in the CMW receives a DNS request from an internal DNS client and passes the DNS request to a name service daemon, the name service daemon resolves the query and provides an answer to the front end daemon, and the front end daemon forwards the answer over the network to the original querying client. (see, also, Ex.1003 at 711-4, 711-5).

Therefore, Dalton teaches determining whether the intercepted DNS request corresponds to an internal host, which can be a WWW server. Dalton also teaches that the CMW determines whether the DNS request is requesting access to an internal host based on whether there is a DNS record for the domain name in the DNS request at the SYSTEM INSIDE level.

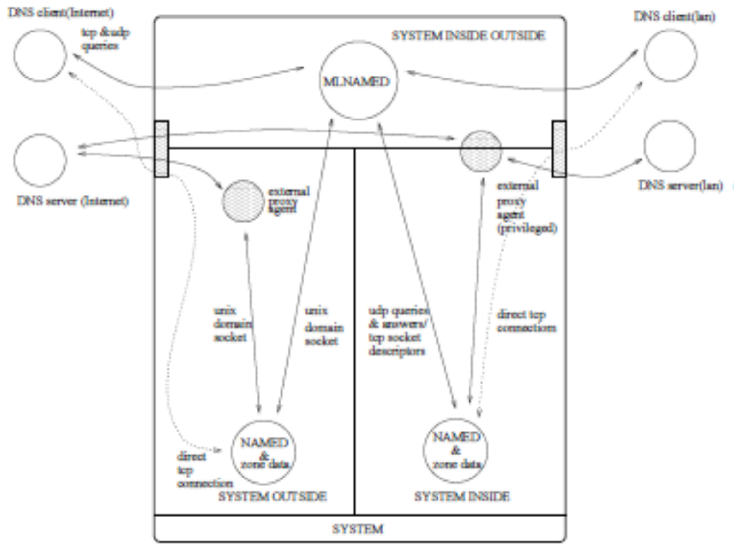
[1.2] (ii) when the intercepted DNS request does not correspond to a secure server, forwarding the DNS request to a DNS function that returns an IP address of a nonsecure computer, and	*when the intercepted DNS request does not correspond to a secure server, forwarding the DNS request to a DNS function that returns an IP address of a nonsecure computer:* See the arrow in above Figure 2 from the name service daemon in the "INSIDE" box to the "OUTSIDE" box and then out to the DNS query "[T]he front end daemon would be configured … to proxy packets at the request of the SYSTEM INSIDE name service daemon over either the SYSTEM INSIDE or SYSTEM OUTSIDE networks. This *allows the internal name server to query other non-local name servers out on the Internet in order to resolve an external address for an internal client* …." (Ex.1003 at 711-4, emphasis added) Dalton teaches "When an untrusted name service daemon needs to query a non-local name server in order to resolve a client query, it sends a query packet via another Unix domain socket to the external proxy agent at its own level. The external proxy agent checks which network level the packet would need to go out at to reach the non-local name server. If the level is the same as its own then it simply *sends the packet out over the network, and passes any replies it receives back to the untrusted name service daemon.*" (Dalton at 711-5, emphasis added). The name service daemon takes over the communications in order to resolve the client query. As in conventional DNS servers, if the name cannot be resolved by the name service daemon, it is passed to a non-local name server. Upon receiving the corresponding IP address, the name service daemon provides it to the originating client. Thus, Dalton includes forwarding the DNS request to a DNS function that returns an IP address for the domain name to the

	client computer in response to having determined that the DNS request is not requesting access to an internal host.
[1.3] (iii) when the intercepted DNS request corresponds to a secure server, automatically initiating an encrypted channel between the client and the secure server.	*when the intercepted DNS request corresponds to a secure server, automatically initiating an encrypted channel between the client and the secure server:* If the CMW determines that the DNS request is requesting access to an internal host on the internal network, the CMW resolves the IP address for the domain name locally (represented by the arrow in Figure 2 that loops back to the Local Area Network), specifically using the set of DNS data maintained by the name service daemon running at the SYSTEM INSIDE level:  CMW HOST SYSTEM INSIDE OUTSIDE Local Area Network DNS queries OUTSIDE INSIDE Internet Untrusted Name service Daemon Untrusted Name service Daemon SYSTEM Figure 2: CMW multilevel DNS configuration "…the name service daemon running at the SYSTEM INSIDE level holds a full set of DNS data for that zone. Internal querying clients therefore have access to all the DNS records for their zone" (Dalton at 711-3, second column, fourth paragraph). On the LAN, the IP address is all that the requesting client needs to communicate with the internal host. Thus, Dalton teaches responding to the determination that the DNS request corresponds to an internal host by initiating communications between the client and the internal host without user involvement. Dalton does not teach an encrypted channel. However,there were large incentives to create virtual

	private networks over the Internet, rather than rely on local networks joined together by private circuits. Indeed, Kiuchi taught that a closed HTTP-based virtual network would be an ideal substitute for a private circuit connecting branches of a corporation. Kiuchi teaches how to create such a closed HTTP-based virtual network. Kiuchi teaches a DNS server (i.e., the C-HTTP name server) that receives a DNS request and, in response to determining that the DNS request is requesting access to a secure server (one requiring permission), returns an IP address and VPN (virtual private network) resources used for automatically initiating an encrypted channel between a client computer and the secure server: "A client-side proxy asks the C-HTTP name server whether it can communicate with the host specified in the URL. If the name server confirms that the query is legitimate, it examines whether the requested server-side proxy is registered in the closed network and is permitted to accept the connection from the client-side proxy. *If the connection is permitted*, the C-HTTP name server sends the IP address and public key of the server-side proxy and both request and response Nonce values." (Ex.1002 at 65, sec. 2.3(2)). Without user involvement, an encrypted channel is created as initiated by the return of the VPN resources. The client computer (i.e., client-side proxy) responds to the VPN resources from the C-HTTP name server by sending a request for a secure connection to the target computer: "[a] client-side proxy sends a request for a connection to the server-side proxy, which is encrypted using the server-side proxy's public key and contains the client-side proxy's IP address, hostname, request Nonce value and symmetric data exchange key for request encryption." (Ex.1002 at 65, sec. 2.3(3)).

	Thus, in Kiuchi, the encrypted channel is automatically initiated and created in response to a determination that the DNS request is requesting access to a secure target web site. Kiuchi teaches additional security measures relative to the secure server by authenticating requests for IP addresses using cryptographic techniques: "Both the request to and response from the C-HTTP name server are encrypted and certified, using asymmetric key encryption and digital signature technology." (Ex.1002 at 65.) Kiuchi further teaches that the secure C-HTTP name server authenticates queries received from a client: "In C-HTTP, five kinds of security technologies are used. They are: 1) asymmetric key encryption for the secure exchange of data encryption keys between two types of proxies and host information between a proxy and C-HTTP name server, 2) symmetric key encryption for the encryption of C-HTTP encrypted headers and HTP/1.0 requests, 3) electronic signature for the request/response authentication, 4) a one-way hash function for checking data tampering and 5) random key generation technology." (Ex. 1002 at 64, emphasis added.) Replacing Dalton's private, closed LAN with a closed virtual network of the type taught by Kiuchi would involve merely modifying Dalton's CMW to include Kiuchi's VPN establishment functions and connecting the internal hosts to the Internet using the CMW as their proxy. Thus, in the CMW, the name server daemon would add the function of returning the public key and Nonce

	values along with an IP address if the DNS request from the "internal" client computer is requesting access to an "internal" target computer. The internal computers will include VPN establishment functions performed by Kiuchi's client-side proxy and server-side proxy (e.g., implemented respectively in the client and target computers, or implemented respectively in firewall devices protecting the client and target computers as in Kiuchi). Such a modified CMW also would handle requests from the server-side VPN establishment function to authenticate the client computer as in Kiuchi (see Ex.1002 at 65-66). In other words, the "internal" computers would be connectable by encrypted channels created in accordance with the closed network set up by Kiuchi on the Internet. A modified CMW implementing Kiuchi's name server functions performs "automatically initiating an encrypted channel" by returning VPN resources along with an IP address in response to the DNS request, which effectively initiates and creates the encrypted channel, in accordance with Kiuchi.
[13.0] A computer readable medium storing a domain name server (DNS) module comprised of computer readable instructions that, when executed, cause a data processing device to perform the steps of:	*A computer readable medium storing a domain name server (DNS) module comprised of computer readable instructions that, when executed, cause a data processing device to perform the steps of:* As discussed above, Dalton's CMW is a data processing device that stores a domain name server (DNS) proxy module. The CMW necessarily includes a computer readable medium with executable computer readable instructions.
[13.1] (i) determining	[13.1] *determining whether a DNS request sent by a client corresponds to a secure server:*

whether a DNS request sent by a client corresponds to a secure server;	See claim portion [1.1], which is identical except for claim 13 reciting a "DNS request sent by a client" instead of "intercepted DNS request." Figure 3 explicitly shows DNS client (lan) and arrows show DNS requests sent by the clients. Arrow from the front end daemon (MLNAMED) is labeled udp queries:  DNS client(Internet) tcp &udp queries SYSTEM INSIDE OUTSIDE DNS client(lan) MLNAMED DNS server (Internet) external proxy agent external proxy agent (privileged) DNS server(lan) unix domain socket unix domain socket udp queries & answers/ tcp socket descriptors direct tcp connection NAMED & zone data NAMED & zone data direct tcp connection SYSTEM OUTSIDE SYSTEM INSIDE SYSTEM Figure 3: Detailed MLNAMED / NAMED architecture Dalton explicitly teaches that the front end daemon (MLNAMED) of the CMW receives DNS requests: > The main body of the front end daemon (MLNAMED) is implemented as a continuous loop. It sits and waits for incoming DNS queries (both TCP and UDP) and for answers sent back from the untrusted name service daemons that require forwarding over the network to the original querying clients. (Dalton at 711-4)
[13.2] (ii) when the DNS request does not correspond to a secure server,	[13.2] *(ii) when the DNS request does not correspond to a secure server, forwarding the DNS request to a DNS function that returns an IP address of a nonsecure computer:*

forwarding the DNS request to a DNS function that returns an IP address of a nonsecure computer; and	See claim portions [1.2a-1.2b], which are identical.
[13.3] (iii) when the intercepted DNS request corresponds to a secure server, automatically creating a secure channel between the client and the secure server.	[13.3] *when the intercepted DNS request corresponds to a secure server automatically creating a secure channel between the client and the secure server:* See claim portion [1.3], which is identical except that claim 13 recites "creating" and "secure channel" instead of "initiating" and "encrypted channel." Kiuchi states that the client-side proxy and the server-side proxy "communicate with each other using a secure, encrypted protocol." (Kiuchi at 64, Abstract) Just as the '151 patent describes "Use of a DNS Proxy to Transparently Create Virtual Private Networks" by sending a message to a gatekeeper, so too does Dalton/Kiuchi disclose creating a virtual private network when the CMW sends out public key and Nonce values in response to a DNS request. Responsive to these VPN resources the channel is created without user intervention, i.e., automatically.