

EX 1003

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

FORTINET, INC.
Petitioner

v.

Netskope, Inc.,
Patent Owner

INTER PARTES REVIEW OF U.S. PATENT NO. 8,635,697
Case No.: *IPR2026-00031*

DECLARATION OF DR. VIJAY MADISETTI

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
II.	BACKGROUND AND QUALIFICATIONS.....	1
III.	PERSON OF ORDINARY SKILL IN THE ART	5
IV.	MATERIALS CONSIDERED.....	5
V.	GENERAL BACKGROUND	7
A.	The TCP/IP Stack	7
B.	OS Fingerprinting	12
C.	p0f.....	13
1.	<i>Key TCP/IP Header Fields Used for OS Fingerprinting</i>	15
2.	<i>Key TCP/IP Header Fields Used for OS Fingerprinting</i>	17
3.	<i>Operational Workflow</i>	17
4.	p0f v. 2.0.8 (Ex. 1012).....	18
D.	Malware Signatures	26
VI.	OVERVIEW OF THE '697 PATENT.....	26
VII.	CLAIM CONSTRUCTION	29
VIII.	INVALIDITY	29
A.	Prior Art Relied Upon.....	31
1.	U.S. Patent Pub. No. 2010/0161795 to Deridder et al. ("Deridder")	31
2.	U.S. Patent Pub. No. 2008/0086773 to Tuvell et al. ("Tuvell").....	33
3.	KR100765340B1 to Lee ("Lee").....	35

B.	Claims 1-25 are Rendered Obvious by Deridder Over Tuvell and Lee.....	36
1.	Motivation to Combine Deridder with the Combined Malware System of Tuvell and Lee (part 1).....	37
2.	Motivation to Combine the Malware Systems of Tuvell and Lee	38
3.	Motivation to Combine Deridder with the Combined Malware System of Tuvell and Lee (part 2).....	40
4.	Claim 1	42
5.	Claim 2	59
6.	Claim 3	63
7.	Claim 4	66
8.	Claim 5	70
9.	Claim 6	71
10.	Claim 7	72
11.	Claim 8	74
12.	Claim 9	75
13.	Claim 10	78
14.	Claim 11	78
15.	Claim 12	80
16.	Claim 13	81
17.	Claim 14	83
18.	Claims 15 and 24	84
19.	Claims 16-23, 25.....	87
IX.	CONCLUSION	89

I, Vijay K. Madiseti, hereby declare as follows:

I. INTRODUCTION

1. I have been retained as an expert by Fortinet Inc. ("Fortinet") and have been asked to provide my opinions regarding U.S. Patent No. 8,635,697 ("the '697 Patent") (Ex. 1001).

2. I am being compensated for my time at my standard consulting rate. I am also being reimbursed for expenses that I incur during the course of this work. My compensation is not contingent upon the results of my study or the substance of my opinions.

II. BACKGROUND AND QUALIFICATIONS

3. A detailed description of my professional qualifications, including a list of publications, awards, and professional activities, is contained in my curriculum vitae, a copy of which is attached as Appendix A.

4. I am a Professor of Cybersecurity and Privacy in the College of Computing at the Georgia Institute of Technology. I lead several research and educational programs at Georgia Tech in the area of cybersecurity, privacy, artificial intelligence, blockchain, digital signal processing, embedded computing systems, chip design, wireless and telecom systems, and systems engineering.

5. I am an IEEE Fellow and, in 2006, I was awarded the 2006 Frederick Emmons Terman Medal by the American Society of Engineering Education (ASEE) and HP Corporation for my contributions to electrical engineering.

6. I have authored or co-authored several books and have been an active consultant to industry and various research laboratories (including MIT Lincoln Labs and JHU Applied Physics Laboratory). I have founded three companies in the areas of embedded software, military chipsets, and wireless communications.

7. I have taught classes at Georgia Tech in the College of Computing and also within the College of Engineering for over thirty years. Topics include information security, cybersecurity incident response, cloud security, embedded software systems, wireless and networking, digital signal processing, speech, audio, video, and image processing, digital signal processing hardware and software, and advanced computing environments.

8. I have been active in the area of computer and information security and protection in the networked environment since the late 1980s, starting with my work on "GAFFES: A Design of A Globally Distributed File System" (EECS Technical Report, UCB/CSD-87-361, June 1987), which described early work on security, authentication, and replication in the network context. I have also published papers in the area of coding theory for secure storage, communications, and noise immunity in the context of storage networks (See "Constrained

Multitrack RLL Codes for the Storage Channel," IEEE Transactions on Magnetics, Vol 31, Issue 3, 1995, for instance). I have also developed algorithms for detection of erroneous (or false) information that can be introduced and propagated into computer networks, and developed a preemptive algorithm called WOLF that has been efficient in limiting the propagation by rolling back the effects of incorrect messages within a network. (See "WOLF: A Rollback Algorithm for Optimistic Distributed Simulation Systems," 1988).

9. More recently, I have co-authored several books: "Cloud Computing: A Hands-On Approach" (2013) and "Internet of Things: A Hands-On Approach" (2014), which discuss recent developments in security for cloud-based networks and networked systems, and these books are already being used as prescribed textbooks by several universities around the world.

10. I have co-authored an invited paper in the Proceedings of the IEEE, the flagship publication of IEEE, entitled "Next Generation Applications on Cellular Networks: Trends, Challenges and Solutions," (2012) in collaboration with engineers from Ericsson, that describes and surveys recent developments in security and authorization in the context of cellular networks.

11. I have consulted to the industry since 2004 timeframe in the areas of Security Certification & Accreditation (C&A) of Computer Systems & Networks in prescribing and evaluating compliance to Information Security Standards

prescribed by US National Institute of Standards & Technology (NIST), specifically NIST 800-30/37, and also in the area of billing fraud and security threats in cellular communications networks.

12. I hold a Bachelor of Technology (Honors) in Electrical & Electronic Communications Engineering from the Indian Institute of Technology (IIT), and a Ph.D. in Electrical Engineering and Computer Sciences (EECS) from the University of California at Berkeley. I have published extensively, with about 120 technical publications and a dozen books in the area of cybersecurity, cloud computing, big data analytics, signal processing, and communications systems. Of these, several relate directly to the design of efficient protocols & codes for information protection & storage, information security in wireless networks, and secure and efficient software for distributed networked systems.

13. My CV outlines my papers (over 30+ papers in the area of cybersecurity) and patents (over 70 issued US patents in the area of security, blockchain, and software systems), and provides additional information about my experience in this area.

14. I have worked as an expert in several legal matters in the prior four years, as identified on my CV, and several of these relate to information security, authentication, and associated protocols.

III. PERSON OF ORDINARY SKILL IN THE ART

15. I have been asked to provide my opinion regarding the qualifications of a person having ordinary skill in the art ("POSITA") with respect to the '697 Patent. I understand that the POSITA is a hypothetical person who is presumed to have known the relevant art at the time of the invention, which I understand here to be sometime at or around the August 2004 filing date of the '697 Patent. I understand that factors that may be considered in determining the level of ordinary skill in the art may include: (a) the type of problems encountered in the art; (b) prior art solutions to those problems; (c) the rapidity with which innovations are made; (d) the sophistication of the technology; and (e) the educational level of active workers in the field.

16. In my opinion, the POSITA relevant to the '697 Patent would have at least an undergraduate degree in computer science or computer engineering. In addition, the POSITA would have at least one year of experience in computer security or networking, or at least 3 years of experience working on computer networking or computer security, even without a formal degree.

IV. MATERIALS CONSIDERED

17. In reaching the conclusions described in this declaration, I have considered the documents and materials cited herein. My opinions are also based

upon my education, training, research, knowledge, and personal and professional experience. The materials review include at least:

- a.** U.S. Patent No. 8,635,697 ("the '697 patent") – Ex. 1001;
- b.** File History of the '697 patent – Ex. 1002;
- c.** U.S. Patent Pub. No. 2008/0086773 to Tuvell et al. ("Tuvell") – Ex. 1004;
- d.** The Certified Translation of KR100765340B1 to Lee ("Lee") – Ex. 1005;
- e.** U.S. Patent Pub. No. 2010/0161795 to Deridder et al. ("Deridder") – Ex. 1006;
- f.** Request for Comments: 2616, Hyper Text Transfer Protocol – HTTP/1.1, 1999 ("RFC 2616") – Ex. 1007;
- g.** Request for Comments: 793, Transmission Control Protocol, September 1981 ("RFC 793") – Ex. 1008;
- h.** A Robust Classifier for Passive TCP/IP Fingerprinting, by Beverly – Ex. 1009;
- i.** Index of _p0f3_releases_old_2.x, available at <https://lcamtuf.coredump.cx/p0f3/releases/old/2.x/> - Ex. 1010;
- j.** U.S. Patent No. 7,231,665 to McArdle et al. ("McArdle") – Ex. 1011; and

k. The source code for p0f, v. 2.0.8 – Ex. 1012.

V. GENERAL BACKGROUND

18. The '697 patent was filed on April 8, 2011, and claims priority to a provisional application dated March 29, 2011. Ex. 1001. For the purposes of my declaration, I have been asked to assume a priority date of March 29, 2011. I render no opinion as to whether the '697 patent is entitled to that date. My opinions rely on this background as technical knowledge that would be possessed by a person of ordinary skill in the art (POSITA) at the time of the '697 patent.

A. The TCP/IP Stack

19. The TCP/IP stack serves as a conceptual framework for understanding how data is transmitted between computers over a network, such as the Internet. As an example, when a client requests a webpage from a server, that request passes through four distinct layers: (1) the application layer, (2) the transport layer, (3) the network layer, and (4) the physical layer. Although various protocols can operate at each of these layers, I limit the discussion here to only those protocols pertinent to the Challenged Claims.

20. **Application layer.** When a website is requested, it is generally done using HTTP (i.e., Hypertext Transfer Protocol), which specifies how the transmitted data should be formatted so that the application—such as a web browser—can correctly render the webpage. Since the 1990s, HTTP has included

a "User-Agent" field that allows the client to communicate certain details about itself to the server. This field typically contains information such as the type of browser and the operating system in use, which helps ensure the data is displayed appropriately by the client's browser. This was well-known to POSITAs at least by the 1990s, and definitely before 2000. Below is a description of that field from the HTTP specification:

14.43 User-Agent

The `User-Agent` request-header field contains information about the user agent originating the request. This is for statistical purposes, the tracing of protocol violations, and automated recognition of user agents for the sake of tailoring responses to avoid particular user agent limitations. User agents SHOULD include this field with requests. The field can contain multiple product tokens (section 3.8) and comments identifying the agent and any subproducts which form a significant part of the user agent. By convention, the product tokens are listed in order of their significance for identifying the application.

Example:

```
User-Agent = "User-Agent" ":" 1*( product | comment )
```

```
User-Agent: CERN-LineMode/2.15 libwww/2.17b3
```

Ex. 1007 at § 14.43 (RFC2616).

21. One of the inputs in the User-Agent field is the "product," which may include the OS of the sender.

3.8 Product Tokens

Product tokens are used to allow communicating applications to identify themselves by software name and version. Most fields using product tokens also allow sub-products which form a significant part of the application to be listed, separated by white space. By convention, the products are listed in order of their significance for identifying the application.

```
product          = token [ "/" product-version ]
product-version = token
```

Examples:

```
User-Agent: CERN-LineMode/2.15 libwww/2.17b3
Server: Apache/0.8.4
```

Product tokens SHOULD be short and to the point. They MUST NOT be used for advertising or other non-essential information. Although any token character MAY appear in a product-version, this token SHOULD only be used for a version identifier (i.e., successive versions of the same product SHOULD only differ in the product-version portion of the product value).

Id. at § 3.8.

22. Put simply, the HTTP protocol communicates the client's operating system through the User-Agent field. Ex. 1007 at 14.43 (RFC 793). This allows the server to tailor its response—for instance, sending different content to a Chrome browser running on a Windows laptop versus a Safari browser on an iPhone. In my experience, this was well-known by anyone familiar with computer networking. Indeed, this type of thing would have been taught to undergraduates in any computer networking class by 2003. I myself taught this to some undergraduate students. I have also seen the OS included in this field in network traces prior to 2003.

23. **Transport Layer.** To facilitate transmission, data sent from the server is generally divided into smaller packets. The transfer of these packets when delivering a webpage is typically handled by TCP (Transmission Control Protocol). TCP is a connection-oriented protocol, meaning the client and server

must first establish a logical connection. This setup allows for error checking, such as detecting lost or corrupted packets, and ensures that all data arrives correctly. Establishing the connection involves a three-way handshake, during which the client and server synchronize packet sequence numbers. This synchronization helps the recipient reassemble packets in the correct order and identify any that are missing.

3.4. Establishing a connection

The "three-way handshake" is the procedure used to establish a connection. This procedure normally is initiated by one TCP and responded to by another TCP. The procedure also works if two TCP simultaneously initiate the procedure. When simultaneous attempt occurs, each TCP receives a "SYN" segment which carries no acknowledgment after it has sent a "SYN". Of course, the arrival of an old duplicate "SYN" segment can potentially make it appear, to the recipient, that a simultaneous connection initiation is in progress. Proper use of "reset" segments can disambiguate these cases.

Several examples of connection initiation follow. Although these examples do not show connection synchronization using data-carrying segments, this is perfectly legitimate, so long as the receiving TCP doesn't deliver the data to the user until it is clear the data is valid (i.e., the data must be buffered at the receiver until the connection reaches the ESTABLISHED state). The three-way handshake reduces the possibility of false connections. It is the

implementation of a trade-off between memory and messages to provide information for this checking.

The simplest three-way handshake is shown in figure 7 below. The figures should be interpreted in the following way. Each line is numbered for reference purposes. Right arrows (-->) indicate departure of a TCP segment from TCP A to TCP B, or arrival of a segment at B from A. Left arrows (<--), indicate the reverse. Ellipsis (...) indicates a segment which is still in the network (delayed). An "XXX" indicates a segment which is lost or rejected. Comments appear in parentheses. TCP states represent the state AFTER the departure or arrival of the segment (whose contents are shown in the center of each line). Segment contents are shown in abbreviated form, with sequence number, control flags, and ACK field. Other fields such as window, addresses, lengths, and text have been left out in the interest of clarity.

TCP A		TCP B
1. CLOSED		LISTEN
2. SYN-SENT	--> <SEQ=100><CTL=SYN>	--> SYN-RECEIVED
3. ESTABLISHED	<-- <SEQ=300><ACK=101><CTL=SYN,ACK>	<-- SYN-RECEIVED
4. ESTABLISHED	--> <SEQ=101><ACK=301><CTL=ACK>	--> ESTABLISHED
5. ESTABLISHED	--> <SEQ=101><ACK=301><CTL=ACK><DATA>	--> ESTABLISHED

Basic 3-Way Handshake for Connection Synchronization

Figure 7.

Ex. 1008 at §3.4

24. **Network Layer.** This layer is responsible for routing and for identifying both the source and destination of data packets. In other words, it establishes the network addresses used to distinguish between devices connected to the internet. The protocol most commonly used at this layer is IP (Internet Protocol), which is the origin of the term "IP address."

25. **Physical Layer.** The physical layer includes the specific physical medium over which data is sent. Common examples of the physical layer include Wi-Fi, Ethernet, and cellular transmissions (3G, 4G, 5G).

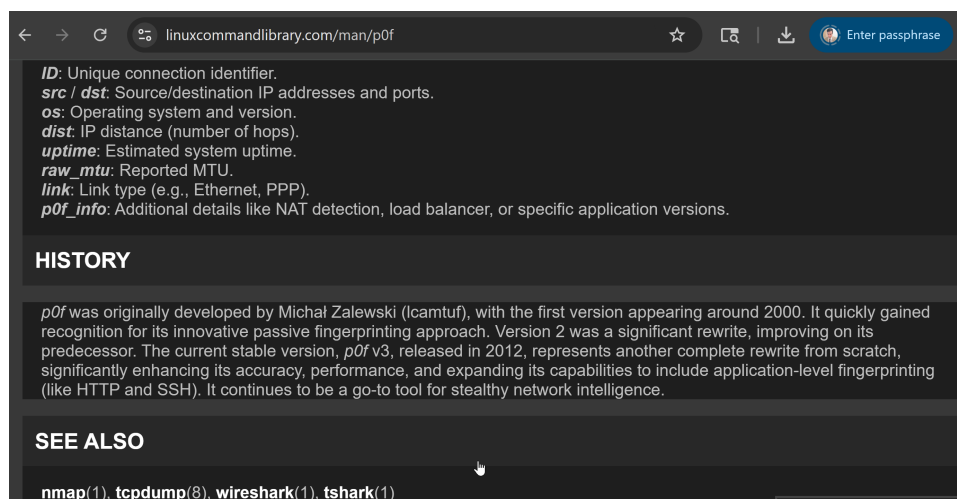
B. OS Fingerprinting

26. OS fingerprinting based on network traffic has been around since the 1990s. The User-Agent field mentioned earlier was introduced prior to the release of HTTP/1.1 in June 1999. For a POSITA, it would have been a straightforward and logical step to examine the User-Agent field to identify the client's operating system. I am aware of this happening and, indeed, security concerns surrounding malicious actors knowing the OS of a device were well-known before the priority date of the patent. *E.g.*, Ex. 1011, 1:26–2:14 (McArdle).

27. Additionally, a widely adopted tool known as nmap, which dates back to 1998, utilizes active fingerprinting techniques by probing open ports on a device. Ex. 1009, 2 (Beverly). Another commonly used open-source utility, p0f, relied on analyzing TCP header signatures—including those in the three-way handshake—to passively infer the client's OS. p0f had reached version 2.0.8 by 2006. Ex. 1010; Ex. 1009, 2 (Beverly); Ex. 1011, 1:26–2:14 (McArdle). I am aware of numerous other tools and studies that also existed for identifying a client's operating system using a variety of techniques. *See, e.g.*, Ex. 1009 (Beverly); Ex. 1011, 1:26–2:14 (McArdle).

C. p0f

28. P0f (Passive OS Fingerprinting) was developed by Michal Zalewski in 2000 as a useful tool for identifying operating systems (OS) without sending any packets to the target system and integrated into many Linux operating system distributions. Unlike active fingerprinting tools like Nmap, p0f passively observed network traffic and analyzed subtle differences in TCP/IP stack implementations to identify the remote operating system. (See <https://seclists.org/bugtraq/2000/Jun/141>, June 10, 2000). It is also available as part of Fedora and Kali Linux projects (See <https://koji.fedoraproject.org/koji/buildinfo?buildID=5404>, May 2, 2007, and <https://koji.fedoraproject.org/koji/packageinfo?packageID=2713>), and <https://linuxcommandlibrary.com/man/p0f>). Though I refer here to the 2000 version, I have also considered version 2.0.8, which was the operative version in 2006. My analysis here also applies to version 2.0.8.



29. Additional recognition of Michal Zalewski's work came through his own publication "Strange Attractors and TCP/IP Sequence Number Analysis" (March 19, 2001) located at <https://lcamtuf.coredump.cx/oldtcp/tcpseq.html>. Although focused on TCP sequence number analysis, this paper described how TCP/IP implementation differences could be leveraged for operating system identification.

30. On October 18, 1998, Fyodor (Gordon Lyon), the creator of Nmap, published "Remote OS Detection via TCP/IP Stack Fingerprinting" (Also published in Phrack Magazine, volume 10, issue 54 at URL <https://phrack.org/issues/54/9>), at <https://nmap.org/nmap-fingerprinting-article.txt>. These articles identify several OS fingerprinting tools that are comparable to p0f.

31. Lippmann et al. published "Passive Operating System Identification from TCP/IP Packet Headers" (2003) at the Workshop on Data Mining for Computer Security (Link at <https://www.ll.mit.edu/r-d/publications/passive-operating-system-identification-tcpip-packet-headers>). This paper built upon p0f's methodologies (Reference [14] in this paper), expanding the analysis of TCP/IP header features for OS identification and citing p0f as the foundation for passive fingerprinting research.

32. In its original version from 2000 (Ex. 1013), p0f relied on analyzing the information contained in TCP headers of packets sent during normal

communication. The passive nature of p0f made it undetectable to the target system, as it did not generate any additional or unusual traffic. This was particularly valuable for security professionals who needed to gather intelligence without alerting the subject system.

33. According to the p0f README file included in the repository, the original version from 2000 focused primarily on analyzing SYN packets, those initial packets used to establish TCP connections. The fundamental premise was that different operating systems implemented their TCP/IP stacks with subtle variations, creating distinctive "fingerprints" that could be observed in the packet headers. From a technical standpoint, p0f uses libpcap for packet capture, setting up appropriate Berkeley Packet Filters (BPF) to capture only relevant TCP packets. For each captured packet, p0f extracts the key TCP/IP header fields, normalizes them (e.g., calculating original TTL values), and compares the resulting fingerprint against its database of known signatures.

1. *Key TCP/IP Header Fields Used for OS Fingerprinting*

34. P0f version 1 (circa 2000) primarily analyzed the following TCP/IP header fields to identify operating systems:

- a. Window Size (WSS): This field controls TCP/IP performance by specifying the maximum amount of data to be sent without acknowledgment. Different operating systems used different

approaches for setting this value—some used fixed values, others calculated it as a multiple of MSS (Maximum Segment Size) or MTU (Maximum Transmission Unit).

- b. Time To Live (TTL):** This field prevents packets from looping indefinitely through networks. Different operating systems set different initial TTL values (commonly 32, 64, 128, or 255). P0f would analyze the observed TTL and estimate the initial value, considering that the TTL decreases by one with each network hop.
- c. Don't Fragment (DF) Flag:** Some operating systems set this flag to implement Path MTU Discovery, while others did not. This created another distinguishing characteristic.
- d. Maximum Segment Size (MSS):** While often network-dependent, some operating systems used distinctive MSS values that could help with identification.
- e. TCP Options and Their Order:** Different operating systems implemented different TCP options (like window scaling, selective ACK, timestamps) and arranged them in different orders within the packet. This sequence became a powerful identifier.

2. Key TCP/IP Header Fields Used for OS Fingerprinting

35. According to the repository, p0f stored these fingerprints in a database file (p0f.fp) with entries formatted like this:

```
'''  
www:ttt:D:ss:OOO...:OS:Version:Subtype:Details  
'''
```

Where:

- `www` - Window size (could be a specific value, wildcard, or multiplier)
- `ttt` - Initial TTL
- `D` - Don't fragment bit (0 - not set, 1 - set)
- `ss` - Overall SYN packet size
- `OOO` - Option value and order specification
- `OS` - OS genre (Linux, Solaris, Windows)
- `Version` - OS Version (2.0.27 on x86, etc.)
- `Subtype` - OS subtype or patchlevel
- `details` - Generic OS details

For example, a signature for Windows XP might look like:

```
'''  
65535:128:1:48:M*,N,N,S:Windows:XP:SP1:Windows XP SP1  
'''
```

36. This would indicate a window size of 65535, initial TTL of 128, DF bit set, packet size of 48 bytes, and TCP options including MSS (wildcard value), two NOP options, and SACK permitted.

3. Operational Workflow

37. When p0f captured a packet in 2000, the workflow proceeded as follows:

- a. Capture the incoming TCP packet with SYN flag set

- b.** Extract the key fields from the TCP/IP headers
- c.** Normalize the observed TTL (estimate the initial value)
- d.** Create a fingerprint string from the observed values
- e.** Compare this string against the database of known fingerprints
- f.** Return the closest match, identifying the operating system (OS)

38. For packets where the TTL had been decremented by routers along the path, p0f would calculate the original TTL by rounding up to the nearest common value (32, 64, 128, or 255).

39. The p0f approach, as described in p0f.c (<https://seclists.org/bugtraq/2000/Jun/141>, June 2000), using TCP/IP packet stack fingerprinting influenced many subsequent security tools and became a standard technique in network security analysis. The underlying principles established in the original 2000 version of p0f continue to inform passive reconnaissance techniques used by security professionals today.

4. p0f v. 2.0.8 (Ex. 1012)

40. p0f underwent considerable evolution between its original version in 2000 and later versions in 2006, with version 2.0.8 representing a complete rewrite of the codebase. The changes reflect a maturation of passive OS fingerprinting techniques and address limitations in the original implementation.

41. ***Core Architecture and Design Philosophy*** - The original p0f.c from 2000 was relatively simple in its approach, focusing exclusively on SYN packet analysis with a limited set of TCP/IP stack parameters. It established the field of the passive fingerprinting concept, but had a straightforward implementation with minimal complexity.

42. By 2006, p0f version 2 emerged as a comprehensive rewrite, designed to support multiple fingerprinting modes and a significantly expanded set of TCP/IP metrics. As stated in the README document for version 2.0.8: "Version 2 is a complete rewrite of the original v1 code. The main reason for this is to make signatures more flexible, and to implement certain additional checks for very subtle packet characteristics to improve fingerprint accuracy."

43. ***Fingerprinting Metrics*** - One of the main differences lies in the packet metrics analyzed.

44. The 2000 version primarily examined basic TCP/IP header fields:

- a. Window size (WSS)
- b. Time To Live (TTL)
- c. Don't Fragment (DF) flag
- d. Overall SYN packet size
- e. Basic TCP options

45. Version 2.0.8 (2006) expanded this analysis with numerous additional metrics:

- a. Option layout and count check
- b. EOL presence and trailing option data
- c. Unrecognized option handling (TTCP, etc.)
- d. WSS to MSS/MTU correlation checks
- e. Zero timestamp check
- f. Non-zero ACK in initial SYN
- g. Non-zero "unused" TCP fields
- h. Non-zero urgent pointer in SYN
- i. Non-zero second timestamp
- j. Zero IP ID in initial packet
- k. Unusual auxiliary flags
- l. Data payload in control packets
- m. SEQ number equal to ACK number
- n. Zero SEQ number
- o. Non-empty IP options

46. Many of these metrics were pioneered by p0f itself, representing significant innovation in passive fingerprinting techniques, as noted in the

README: "Some of the metrics were so precise I managed to find several previously unknown TCP/IP stack bugs."

47. **Detection Modes** – Version 2.0.8 introduced multiple detection modes, augmenting the 2000 version further:

- a. SYN mode (default): Fingerprinting systems that connect to you
- b. SYN+ACK mode: Fingerprinting systems you connect to
- c. RST+ mode: Fingerprinting systems that reject your traffic
- d. RST+ mode: Fingerprinting systems that reject your traffic

48. This expansion dramatically increased p0f's versatility, allowing passive fingerprinting in various network scenarios. As the README notes: "The current version uses a number of detailed metrics, often invented specifically for p0f, and achieves a very high level of accuracy and detail; it is designed for hands-free operation over an extended period of time."

49. **Fingerprint Database Structure** - The original p0f fingerprint database used a relatively simple format, focusing on the core TCP/IP parameters with limited wildcard support. The 2006 version introduced a much more sophisticated and flexible signature format.

50. According to the README, the newer database supported:

- a. Modulo and wildcard operators for certain TCP/IP parameters

- b. More detailed OS information (genre, version, subtype, details)
- c. Extensive signature validation through collision detection
- d. Separate signature files for different detection modes (p0f.fp, p0fa.fp, p0fr.fp, p0fo.fp)

51. ***Performance Optimizations*** - The 2000 implementation had basic performance characteristics, parsing signatures at runtime, and using straightforward comparison algorithms. Version 2.0.8 included significant performance optimizations:

- a. No more runtime signature parsing
- b. Added BPF pre-filtering
- c. Implemented signature hash lookups

52. ***Advanced Analysis Features (including NAT features)*** – Version 2.0.8 introduced sophisticated analytical capabilities absent from the original:

- a. Auto-detection of DF-zeroing firewalls
- b. Auto-detection of MSS-tweaking NAT and router devices
- c. Media type detection based on MSS
- d. Origin network detection based on ToS/precedence bits
- e. Uptime estimation based on TCP timestamps
- f. Masquerade detection for identifying multiple systems behind NAT

53. The masquerade detection feature was particularly innovative, using various indicators to identify when multiple different systems shared a single IP address as when using a NAT (Network Address Translation) device, such as an access control device, such as a wireless router/NAT. As the README explains: "The algorithm looks over recent (cached) hits and looks for indications of multiple systems being behind a single gateway."

54. ***Integration Capabilities*** – The original p0f was a standalone tool with limited integration options. Version 2.0.8 added a query mode through a local socket interface, enabling easy integration with other software. This feature allowed third-party applications to query p0f for OS fingerprinting results.

55. From the README: "Query mode to enable easy integration with third party software - p0f caches recent fingerprints and answer queries for src-dst combinations on a local stream socket in a easy to parse form."

56. ***Security Enhancements*** – Version 2.0.8 incorporated improved security features, including:

- a. The ability to chroot and drop privileges (with -u option)
- b. *Better input validation and error handling*
- c. Daemon mode for background operation
- d. More robust packet parsing with sanity checks

57. ***Usability Improvements*** – The 2006 version added numerous usability enhancements:

- a. Greppable output option
- b. Host name resolution option
- c. Promiscuous mode switch
- d. Built-in signature collision detector
- e. ToS reporting
- f. Full packet dumps
- g. PCAP dump output
- h. VLAN support

58. ***Signature Database Quality*** – While the original p0f came with a limited set of signatures focused on common operating systems, version 2.0.8 boasted a much more comprehensive and rigorously verified database. The README notes: "Better fingerprint file structure and contents - all fingerprints are rigorously reviewed before being added."

59. ***Fuzzy Matching*** - The 2000 version relied on matching fingerprints. Version 2.0.8 introduced limited fuzzy matching capabilities with the -F option to "deploy fuzzy matching algorithm if no precise matches are found." This improved identification rates when exact matches weren't available.

60. **Community Adoption** - By 2006, p0f had become integrated into numerous security tools and operating systems. The README mentions:

"Portions of this code are used in several IDS systems, some sniffer software; p0f is also shipped with several operating systems and incorporated into an interesting OpenBSD pf hack by Mike Frantzen, that allows you to filter out or redirect traffic based on the source OS."

61. **Conclusion** - The evolution from p0f's 2000 version to the 2006 version 2.0.8 represents a significant advancement in passive fingerprinting technology. The tool transformed from a proof-of-concept with limited capabilities to a sophisticated, multi-mode fingerprinting system with extensive metrics, performance optimizations, NAT integrations for gateways, and integration capabilities. The 2006 version established p0f as the standard for passive OS fingerprinting, with innovations that continue to influence network security tools today.

62. As stated in the README document: "P0f was the first (and I believe remains the best) fully-fledged implementation of a set of TCP-related passive fingerprinting techniques." The substantial differences between the 2000 and 2006 versions reflect the rapid evolution of this *groundbreaking security tool* during its first six years of development.

D. Malware Signatures

63. Signature-based methods for detecting malware have been in use since the 1980s. This approach is based on the observation that malicious software frequently includes recognizable patterns—or "signatures"—that can be identified within network traffic. Security devices such as firewalls, which also originated in the 1980s, leverage this technique by scanning for known signatures and, for instance, blocking any traffic that contains them. Similarly, antivirus software from the 1990s also used this same signature-based approach. *E.g.*, Ex. 1014 (U.S. Patent No. 5,319,776).

VI. OVERVIEW OF THE '697 PATENT

64. I have been asked to review and comment on the '697 Patent as to its validity and whether certain prior art grounds render its claim obvious to one of ordinary skill in the art.

65. The '697 patent concerns methods for detecting computers within a network that have been compromised by malware, such as computer viruses. Ex. 1001, 1:17–21. According to the patent, both residential and commercial networks are commonly connected to the internet through a gateway or router that employs network address translation (NAT). *Id.* at 1:25–27. As a result, even if multiple devices are connected to the network behind the router or gateway, "[a]ll traffic emerging from that network will share the same IP address that is assigned to the

gateway or router," since "Internet service providers typically only provide one wide area network (WAN) address" per location. *Id. at* 1:30–34. Consequently, from the perspective of the external network (WAN), "it is difficult to determine which device behind a NAT device is originating a particular stream of traffic." *Id. at* 1:38–40.

66. The patent insists that this limitation as a significant issue. Specifically, according to the patent, when analyzing packets on the WAN side reveals malware originating from a device behind a NAT, the "inability to determine a specific device means that any remediation procedures can not be applied in a focused manner." *Id. at* 1:43–45. To address this supposed problem, the patent proposes extracting additional identifying details from the TCP packets—such as "the malware ID and the OS ID associated with [the] TCP session that generated the alert." *Id. at* 1:53–67.

67. Figure 1, which I reproduced below, depicts the alleged invention.

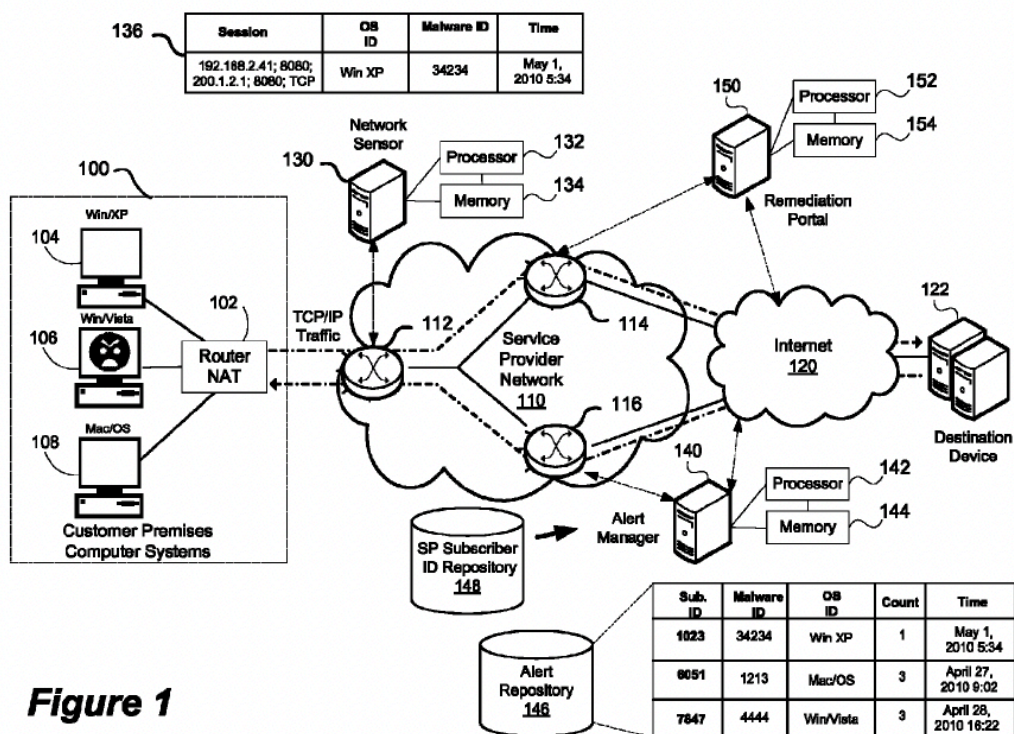


Figure 1

68. In Figure 1, the depicted system includes three computers (104, 106, 108), each running a distinct operating system, connected within a local area network (LAN) 100. This LAN connects to the wider internet through a NAT-enabled router 102. Regardless of which computer sends data, all traffic leaving the NAT shares the same IP address. The traffic then passes through the service provider network 110, which is made up of multiple routers and switches (112, 114, 116). *Id. at* 4:11–33. As the data moves through this network, a network sensor 130 inspects the packets to identify any signs of malware infection on the LAN devices. *Id. at* 4:50–60. Upon detecting malware, the network sensor generates an alert that is forwarded to the alert manager 140, where it is stored in

the alert repository 146. *Id. at 4:60–5:25.* This repository maintains details such as the malware ID and the OS ID, examples being Windows XP, MacOS, or Windows Vista. *Id.*

69. I have also considered the file history in rendering my opinion, which is Exhibit 1002.

VII. CLAIM CONSTRUCTION

70. It is my understanding that, in this IPR, the claims are interpreted consistent with the plain and ordinary meaning of the term as it would have had to a POSITA at the time of the alleged invention.

71. In my opinion, no constructions are necessary because the prior art would disclose or render obvious the claims at issue under any reasonable construction of any term. To the extent Patent Owner attempts to offer specific, narrower constructions, I reserve the right to address those positions.

VIII. INVALIDITY

72. I have been asked to provide opinions addressing the validity of certain claims of the '697 Patent in light of prior art.

73. My understanding is that a claimed invention is unpatentable if a piece of prior art "anticipates" the claimed invention, which I understand to mean that the prior art discloses each and every limitation of the claimed invention.

74. My understanding is that a claimed invention is also unpatentable if the differences between the invention and the prior art are such that the subject matter as a whole would have been "obvious" at the time the invention was made to a POSITA. I have been instructed in this regard that a patent may not be obtained if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a POSITA to which said subject matter pertains. I have further been instructed that a determination of obviousness should consider the scope and content of the prior art, the differences between the claimed invention and the prior art, and the level of ordinary skill in the pertinent art.

75. I have been instructed that rationales that may support a conclusion of obviousness include:

- a. Combining prior art elements according to known methods to yield predictable results;
- b. Simple substitution of one known element for another to obtain predictable results;
- c. Use of known technique(s) to improve similar methods or apparatuses in the same way;
- d. Applying a known technique to a known method or apparatus ready for improvement to yield predictable results;

- e. "Obvious to try"—choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success;
- f. Known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces if the variations are predictable to one of ordinary skill in the art;
- g. Some teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify the prior art reference or to combine prior art reference teachings to arrive at the claimed invention.

76. In my opinion, the references below disclose and suggest claimed features of the '697 Patent, rendering the claims discussed below obvious.

A. Prior Art Relied Upon

1. U.S. Patent Pub. No. 2010/0161795 to Deridder et al. ("Deridder")

77. Deridder was filed December 22, 2009. I was instructed to assume this is the priority date for Deridder and do not have an independent opinion or analysis of that date at this time. Deridder is titled "Apparatus and Method for Multi-User NAT Session Identification and Tracking." Deridder was not considered by the examiner during prosecution of the '697 Patent. Deridder

describes a network device designed to assist in tracking user sessions in environments utilizing network address translation (NAT). Ex. 1006, Abstract. The disclosure notes that OS fingerprinting can be used to differentiate users who access the internet through a NAT, such as a wireless router. However, Deridder points out that earlier approaches have been unable to perform this fingerprinting analysis in real time. *Id.* at ¶¶4–5. To address this, Deridder presents a method for identifying the "OS type" of each device by leveraging information contained within the TCP packets transmitted by that device. *Id.* at ¶47.

78. For instance, a traffic inspection device within the system can utilize the IP and TCP header fields from the initial TCP synchronization (SYN) packet of a connection to create a unique signature, or "fingerprint," of the operating system that generated the packet. *Id.* at ¶31. Deridder further explains that identifying the OS through fingerprinting can be valuable for network devices offering services such as network monitoring, targeted advertising, **malware detection**, and similar functions. *Id.* at ¶22.

79. Figure 1 illustrates an example network setup in Deridder's system, where the NAT 102 manages access to the service provider network 110. *Id.* at ¶21. The traffic inspection device 150 analyzes the traffic originating from the NAT and records data such as the OS fingerprint. *Id.*

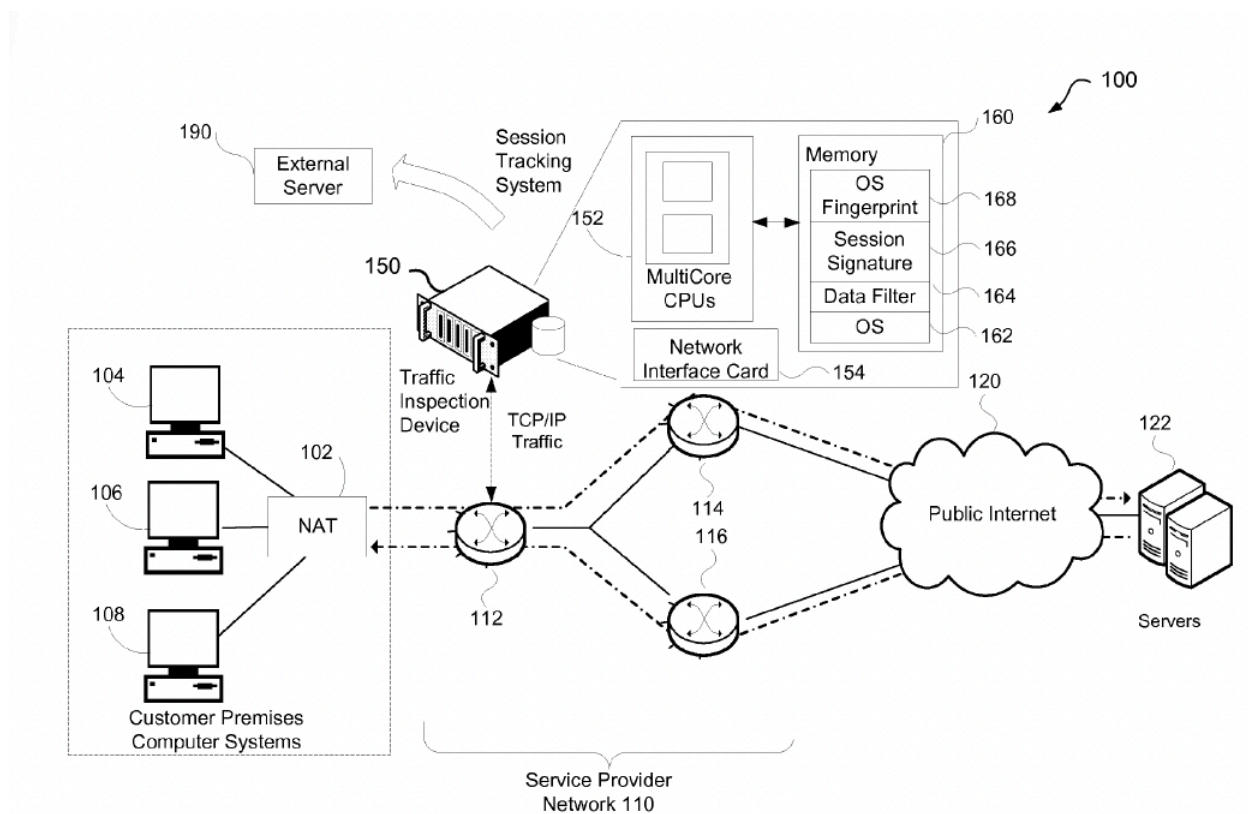


Figure 1

2. U.S. Patent Pub. No. 2008/0086773 to Tuvell et al. ("Tuvell")

80. Tuvell was filed October 9, 2007. I was instructed to assume this is the priority date for Tuvell and do not have an independent opinion or analysis of that date at this time. Tuvell is titled "System and Method of Reporting and Visualizing Malware on Mobile Networks." Tuvell was not considered during prosecution of the '697 Patent during prosecution. Tuvell describes a malware detection system that includes a "network analyzer 202" and a "malware scanner 204," both components of a network management system. Ex. 10 ¶34. This system is illustrated in Figure 3 of the patent.

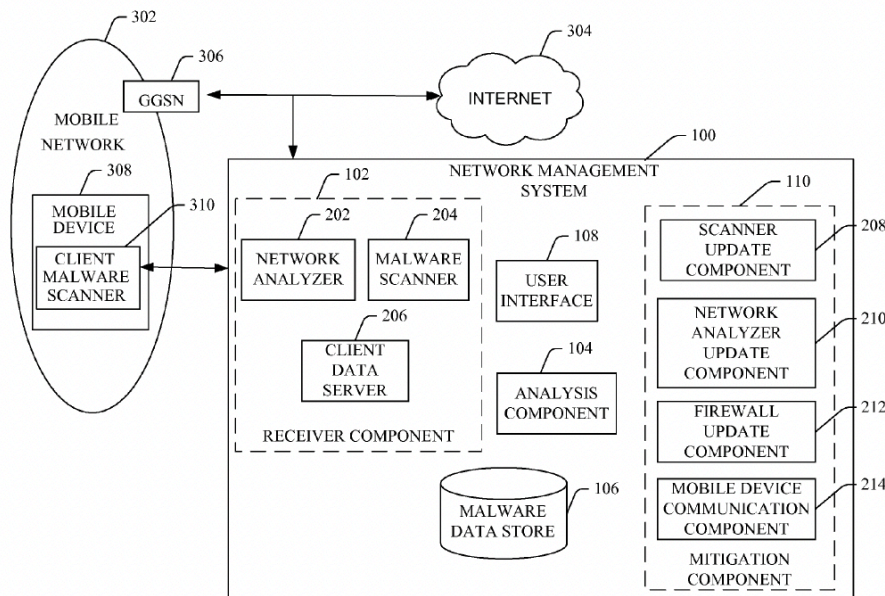


FIG. 3

81. As shown in Figure 3 from Tuvell above, Tuvell's network management system monitors data traveling from the mobile network 302 to the internet 304 through a GGSN 306. According to Tuvell, the network analyzer "intercepts and logs traffic passing over a network or a portion of a network," and in one embodiment, it specifically captures communications between the mobile network and a data network such as the internet. *Id.* at ¶33. The captured packets are then examined by the "malware scanner 204," which identifies both the presence of malware in the data and the addresses of the transmitting and/or receiving mobile devices. *Id.* at ¶34. This analysis is performed in real time. *Id.* Tuvell further states that the analysis may involve techniques such as "malware signature searches" and "hash signature searches." *Id.* at ¶35.

82. Tuvell also states that the "malware data obtained by network analyzer 207 and client data server 206 includes device-specific information." *Id.* The system correlates the malware data received by the network analyzer 202 with data from the mobile network to determine which mobile device transmitted the infected packet or packets. *Id. at* ¶36. Among the information received about the mobile device, Tuvell notes that the device's operating system is included. *Id. at* ¶49. The results of the malware analysis are stored in the malware data store 106 and/or displayed to operators through the user interface 108. *Id. at* ¶37. According to Tuvell, the malware data store contains details such as the virus name, infected file name, file size, infected packet size, and packet number. It also retains source-related information, including the source IP, source port, and even a source identifier like a phone number. *Id. at* ¶52.

3. KR100765340B1 to Lee ("Lee")

83. Lee is a Korean Patent that was published April 26, 2006. I was instructed to assume this is the priority date for Lee and do not have an independent opinion or analysis of that date at this time. Lee is titled "Virtual In-Line Network Security Method." Lee was not before the examiner during prosecution of the '697 Patent. Lee similarly describes an inline network security system—such as a firewall—that monitors and inspects network traffic for signs of malware. Ex. 1005 at ¶13. For instance, Lee explains that the payload of each

packet can be compared against entries in a malware database. If a match is found, the system logs the event and assesses whether the client's traffic is malicious. Ex. 1005 at ¶30. As part of this logging process, Lee further discloses identifying and recording the client's "operating system (OS)." Ex. 1005 at ¶48.

B. Claims 1-25 are Rendered Obvious by Deridder Over Tuvell and Lee

84. In this section, I will explain why the teachings or combinations of teachings from the following references either disclose or render obvious each of the Challenged Claims:

Name	Patent/Title	Exhibit	Priority Date
Deridder	US 2010/0161795	1006	December 22, 2009
Tuvell	U.S. 2008/0086773	1004	October 9, 2007
Lee	KR100765340B1	1005	April 26, 2006

- a. In my opinion, it would have been obvious for a POSITA to combine the NAT-based system in Deridder with the malware detection systems in Tuvell and Lee. I first address why it would have been obvious for a POSITA to add a malware detection system like those in Tuvell and Lee to the NAT-based system in Deridder. I next address why it would have been obvious for a POSITA to combine the malware systems from Tuvell and Lee.

- b. I have been instructed to assume that each of these references qualifies as prior art. I do not have an independent opinion or analysis of those dates at this time.
- c. As I explain below, the Challenged Claims are invalid based upon the following prior art ground:
 - i. Ground 1: Challenged Claims 1-25 are rendered obvious by Deridder over Tuvell and Lee

1. Motivation to Combine Deridder with the Combined Malware System of Tuvell and Lee (part 1)

85. To begin with, Deridder explicitly states that its system would be beneficial for services "such as network monitoring, targeted advertising, malware detection etc." Ex. 1006 at ¶22. In my opinion, this acknowledgment indicates that a POSITA would have understood the value of integrating malware detection capabilities on the service provider side of a NAT. In such scenarios, Deridder also recognizes that its OS fingerprinting method would be useful in distinguishing individual mobile devices connecting through the NAT—particularly because all devices behind the NAT appear to share the same network address from the service provider's perspective. This would have been obvious to a POSITA, particularly in light of Deridder's contemplation that it be used in such a way.

86. Accordingly, it would have been obvious to a POSITA to incorporate malware detection functionality into the service provider network and to utilize

Deridder's OS fingerprinting technique, given Deridder's express acknowledgment of its applicability to malware detection. Ex. 1006 at ¶22. A POSITA would have consulted other references in the field for guidance on the implementation of a malware detection system. Tuvell and Lee are two such references that describe systems for detecting malware. Moreover, combining the Combined Malware System of Tuvell and Lee (which I explain next) with Deridder's system would have been an obvious step, as both Tuvell and Lee emphasize determining the client device's operating system. Ex. 1004 at ¶49; Ex. 1005 at ¶55 ("operating system (OS) of the aforesaid extracted node").

2. Motivation to Combine the Malware Systems of Tuvell and Lee

87. In my opinion, it would have been obvious to a POSITA to combine the teachings of Tuvell and Lee, as both references address similar objectives within the field of network security by intercepting and monitoring network traffic. According to Tuvell, its system "provides a comprehensive means for collecting, reporting, and providing visual depictions of information regarding the propagation and effect of worms, viruses and other malware on a network." Ex. 1004 at ¶9. Likewise, Lee describes a "virtual in-line network security mechanism" operating as a general-purpose network device after being connected to the network, distinguishing it from traditional configurations. Ex. 1005 at ¶33. Specifically, Tuvell discloses network components including a network analyzer and malware

scanner that "intercepts and logs traffic passing over a network or a portion of a network." Ex. 1004 at ¶33. Similarly, Lee teaches implementation of its system within a network device, such as a firewall, which a POSITA would recognize as intercepting and logging network traffic flowing into or out of the network. Ex. 1005 at ¶13.

88. It is my opinion that a POSITA would have had motivation to combine the system disclosed in Tuvell with that of Lee because Tuvell already teaches correlating network traffic data with the operating system type. Ex. 1004 at ¶49. A POSITA would therefore appreciate the value of linking OS type to the virus name, given that many viruses target specific operating systems. Furthermore, in my opinion, a POSITA would have been motivated to identify the OS type independently—without relying on input from the mobile devices—to reduce network overhead and enhance the robustness of the network analyzer. In essence, Tuvell acknowledges the usefulness of associating OS type with network traffic, prompting a POSITA to seek methods for determining the OS type.

89. Consequently, it is my opinion that a POSITA would have been motivated to seek a system capable of determining the operating system based on network traffic, and Lee discloses one such approach. Specifically, Lee states that one of the data points to extract from network packets is the "operating system (OS) of the aforesaid extracted node." Ex. 1005 at ¶55. Additionally, given a

POSITA's familiarity with the HTTP protocol, it is my opinion that it would have been understood that OS information can be obtained from the User-Agent field, where clients typically reveal their OS type. Ex. 1007 at 14.43, 3.8.

90. Accordingly, a POSITA would have been motivated to combine the system disclosed in Tuvell with the teachings of Lee, and would have had a reasonable expectation of success given that both references describe similar systems operating on comparable network devices.

3. Motivation to Combine Deridder with the Combined Malware System of Tuvell and Lee (part 2)

91. Thus, as I just explained, while Lee at least suggests one approach for obtaining the OS ID—by extracting it from the User-Agent field, as discussed further below—a POSITA would have recognized that this method is not always reliable, as some clients may omit OS information from the User-Agent string. This would have motivated a POSITA to explore alternative techniques for determining the OS type when such information is absent. Deridder offers one such alternative by performing OS fingerprinting using characteristics found in "IP and TCP header fields from the initial TCP synchronization (SYN) packet of a connection to provide a unique signature or 'fingerprint' of the OS that generated the packet." Ex. 1006 at ¶31.

92. Moreover, a POSITA would have had a reasonable expectation of success in combining these references, as the combine system of Tuvell and Lee

that I explain next—like the traffic inspection device in Deridder—operate on network components that observe all traffic traversing the network. For instance, Tuvell describes monitoring data as it passes through the GGSN in a cellular network, which functions as part of the service provider network—consistent with the network model described in Deridder. Similarly, Deridder's traffic inspection device monitors traffic passing through the NAT on the service provider side. Furthermore, both the GGSN and NAT provide network address translation functionality for the network. Ex. 1004 at ¶33; Ex. 1005 at ¶13; Ex. 1006 at ¶21, Fig. 1. It is therefore my opinion that it would have been obvious to a POSITA that the combined malware detection system of Tuvell and Lee could be combined with the NAT-based system in Deridder with very minor changes to identify devices in a local area network instead of in a cellular network.

93. Additionally, in my opinion, a POSITA would have been motivated to employ both OS fingerprinting and analysis of the User-Agent field as complementary methods to verify the accuracy of the identified OS ID for a given mobile device. It is my opinion that a POSITA would understand that each technique—on its own—could be susceptible to inaccuracies or failure in certain scenarios. Given that some malware strains are specific to particular operating systems, it would have been obvious to a POSITA that accurately determining the OS is critical for effective malware detection and response.

94. Accordingly, it is my opinion that a POSITA would have been motivated to integrate the malware detection capabilities described in the Tuvell-Lee combination into the NAT-based architecture of Deridder, as Deridder expressly contemplates the application of its system to malware detection scenarios.

4. Claim 1

95. In my opinion, each and every limitation of Claim 1 is rendered obvious by the combination of Deridder in view of Tuvell and Lee.

96. Claim 1 recites: **1[pre]** "A method of network based malware detection in a service provider network, the method comprising." I note that I labeled each of the limitations within each claim for convenience, so I can refer back to them easily. I have been asked to assume the preamble is limiting. It is my opinion that the combination of Deridder with Tuvell and Lee discloses this system.

- a. I rely upon Deridder in combination with Tuvell and Lee to disclose and render obvious this limitation, in combination with the knowledge of a POSITA.
- b. Deridder discloses a NAT-based system with a traffic inspection device **150** in a service provider network used to determine the OS of the computers on the other side of the NAT. This is depicted in Figure 1:

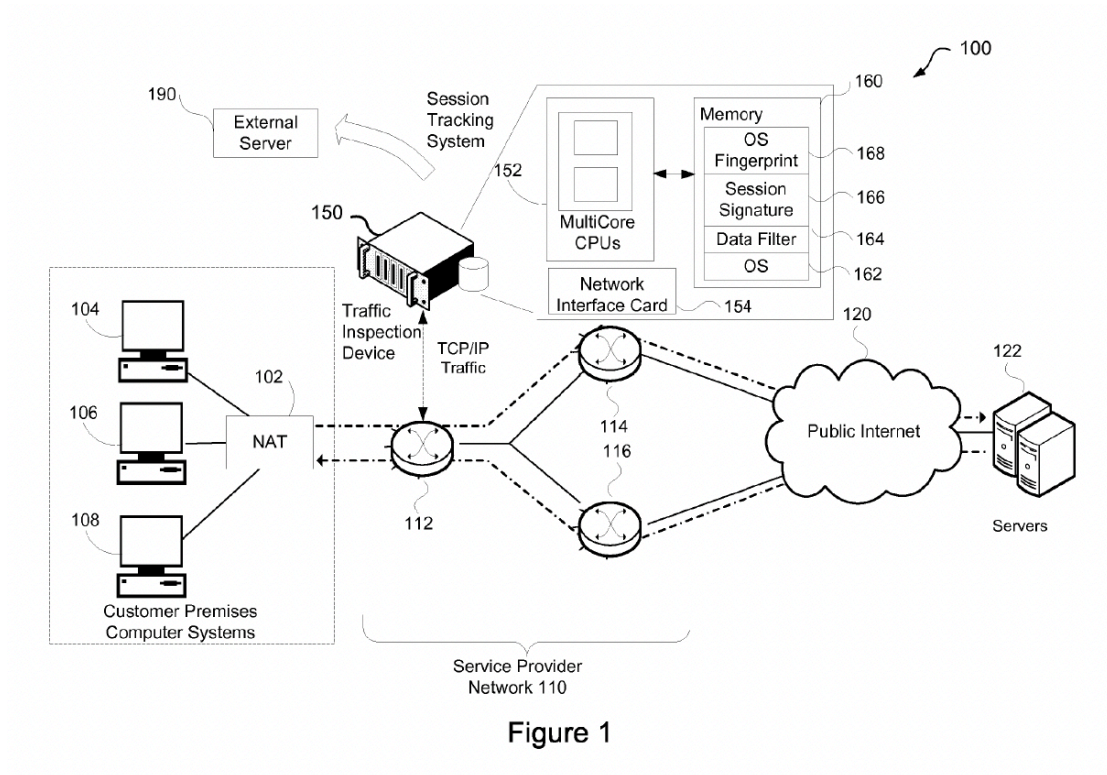


Figure 1

- c. This is the same network architecture depicted in Figure 1 of the '697 patent:

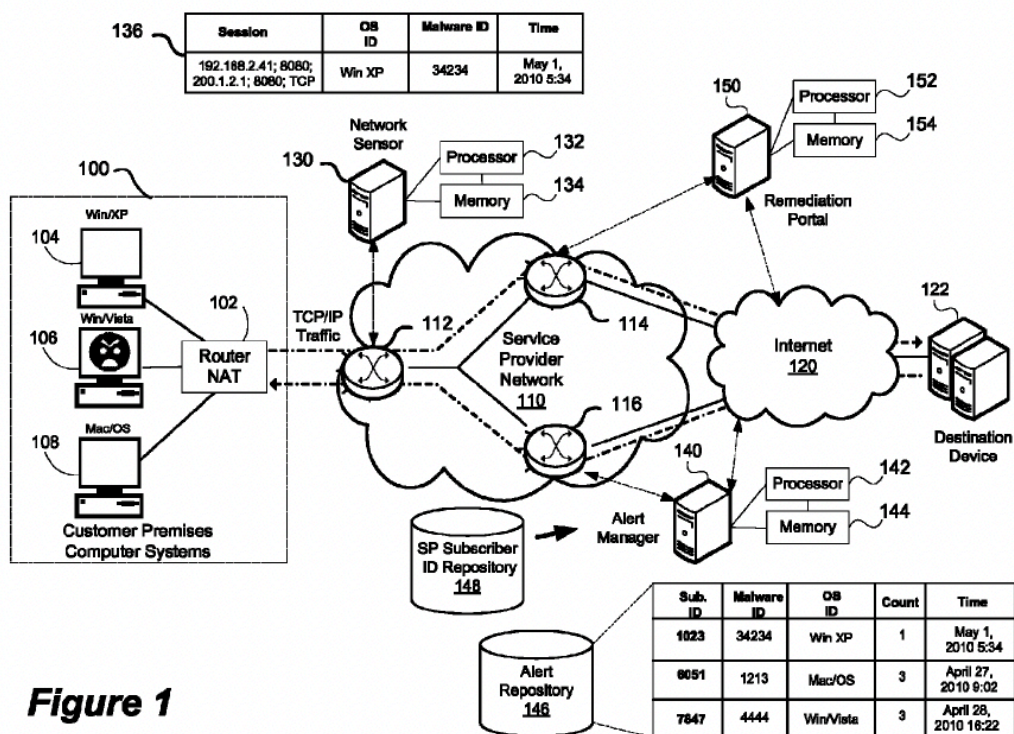


Figure 1

Ex. 1001, Fig. 1.

"FIG. 1 is a schematic representation of a session tracking system 100 for multi-user NAT session identification and tracking in a typical operating environment. The network traffic on the service provider network 110 is passed to a traffic inspection device 150 with a high-speed network interface that is capable of receiving all network packets for computational analysis. The service provider network comprises multiple routing devices, such as 112, 114 and 116 connected to additional networks such as the public internet 120 which external servers 122 are connected to. The session tracking system 100 can be implemented as part of the traffic inspection device (e.g. data filter) 150. The inspection system 150 can be implemented at the edge of the ISP network at router 112 to ensure all traffic between a user computer system (104 to 108) is inspected. Alternatively the inspection device may be located within the service provider network 110 with packets received from the NAT device 102 being copied or forwarded to the inspection device 150. Alternatively, the traffic inspection device 150 may function inline with the network traffic. The system can optionally be operated between a first mode in which NAT session tracking is enabled and a second mode in which NAT session tracking is disabled. When NAT

session tracking is disabled, all traffic to or from a single client Internet Protocol (IP) address is viewed as belonging to a single user or session. When NAT session tracking is enabled, the sessions associated with multiple users (104 to 108) behind a NAT device 102 can be detected, and their traffic can be differentiated." Ex. 1004 at ¶ 21.

d. Deridder explains that the OS determination may be useful for: "for services such as network monitoring, targeted advertising, *malware detection* etc." Ex. 1004 at ¶ 22.

e. Tuvell discloses a network-based malware detection system:

"Turning now to FIG. 2, a network management system or CoreStats 100 is illustrated in greater detail. In an embodiment, the receiver component 102 includes a network analyzer 202 or packet sniffer that monitors network traffic. The network analyzer 202 can be implemented as software and/or hardware that intercepts and logs traffic passing over a network or a portion of a network. In an embodiment, the network analyzer 202 intercepts communications between the mobile network and a data network (e.g., the Internet). In an alternate embodiment, the network analyzer 202 intercepts data within the mobile network. Intercepted or 'sniffed' *data packets are analyzed by a data stream scanner or malware scanner 204 to identify malware present in the data packets and the addresses of the transmitting and/or receiving mobile device*. Data is analyzed in real time packet by packet or stored and analyzed non-linearly. In some instances, the data packets may need to be reassembled in the proper order and the contents extracted before analysis can be done." Ex. 1004 at ¶¶ 33-34.

f. Tuvell positions the network analyzer/malware detection scanner:

"Referring now to FIG. 3, an exemplary deployment of a sample CoreStats system 100 in a network environment is depicted. FIG. 3 illustrates a deployment of CoreStats 100 *between the edge of a mobile network 302 and the Internet 304, although it can also be deployed effectively at various other points in the mobile network 302 depending upon the network topology and desired coverage. The network analyzer 202 monitors and evaluates all traffic going from the mobile network 302 to the public data networks (e.g., the Internet 304) and vice-versa*. The

network analyzer 202 can intercept packets on either side of a firewall (not shown)." *Id. at* ¶43.

g. Tuvell also explains that "laptops and other mobile electronic devices" can be on the network. *Id. at* ¶25.

h. And, I explained earlier that it would have been obvious to combine Tuvell with Lee.

i. In my opinion, a POSITA would have found it obvious to use the malware detection system in the Tuvell-Lee combination I described above on the traffic inspection device in Deridder for the reasons I already discussed above.

97. Claim 1 recites: **1[a]** "receiving one or more transmission control protocol (TCP) packets originating from an access device coupled to the service provider network, the one or more TCP packets defining a TCP session between a computing device coupled to the access device, and a destination coupled to the service provider network."

a. I rely upon Deridder to disclose and render obvious this limitation, in combination with the knowledge of a POSITA.

b. I touched on this already in connection with 1[pre]. Deridder discloses a traffic inspection device **150** that receives TCP packets originating from the NAT, which is coupled to the service provider network. The TCP packets define a TCP session between a computing device behind the

NAT and the NAT, with a destination somewhere in the service provider network:

"Sessions (i.e. end-users on different computer systems (104 to 108)) are modeled as a combination of an operating system (OS) fingerprint and a session signature. The first stage of session detection involves the detection of an OS fingerprint from captured packets that provides a unique signature for a particular operating system. Multiple OS fingerprints emanating from a single source IP address serve to differentiate disparate computers (104 to 108) behind a NAT device 102. The OS fingerprint stage leverages a degree of packet *reassembly to construct the TCP/IP flow and OS fingerprint information* belonging to application-layer messages." Ex. 1006 at ¶ 23.

- c. Deridder explains that examples of NATs are "home wireless routers."

Id. at ¶ 3; *see* Ex. 1001, Fig. 1 ("Router/NAT" 102).

- d. In my opinion, a POSITA would recognize that a TCP session would be established between computer devices 104, 106, and 108 and the NAT. Additionally, it is my opinion that it would be understood by a POSITA that the TCP packets sent from the NAT to the destination correspond to the TCP session occurring between the computer devices and the NAT. This is because NATs function as intermediary devices that perform network address translation.

- e. In my opinion, a POSITA would also recognize that TCP operates as a connection-oriented protocol, and that a TCP session—or connection—is

established and defined through the exchange of one or more TCP packets.

3.4. Establishing a connection

The "three-way handshake" is the procedure used to establish a connection. This procedure normally is initiated by one TCP and responded to by another TCP. The procedure also works if two TCP simultaneously initiate the procedure. When simultaneous attempt occurs, each TCP receives a "SYN" segment which carries no acknowledgment after it has sent a "SYN". Of course, the arrival of an old duplicate "SYN" segment can potentially make it appear, to the recipient, that a simultaneous connection initiation is in progress. Proper use of "reset" segments can disambiguate these cases.

Several examples of connection initiation follow. Although these examples do not show connection synchronization using data-carrying segments, this is perfectly legitimate, so long as the receiving TCP doesn't deliver the data to the user until it is clear the data is valid (i.e., the data must be buffered at the receiver until the connection reaches the ESTABLISHED state). The three-way handshake reduces the possibility of false connections. It is the

implementation of a trade-off between memory and messages to provide information for this checking.

The simplest three-way handshake is shown in figure 7 below. The figures should be interpreted in the following way. Each line is numbered for reference purposes. Right arrows (-->) indicate departure of a TCP segment from TCP A to TCP B, or arrival of a segment at B from A. Left arrows (<--), indicate the reverse. Ellipsis (...) indicates a segment which is still in the network (delayed). An "XXX" indicates a segment which is lost or rejected. Comments appear in parentheses. TCP states represent the state AFTER the departure or arrival of the segment (whose contents are shown in the center of each line). Segment contents are shown in abbreviated form, with sequence number, control flags, and ACK field. Other fields such as window, addresses, lengths, and text have been left out in the interest of clarity.

TCP A		TCP B
1. CLOSED		LISTEN
2. SYN-SENT	--> <SEQ=100><CTL=SYN>	--> SYN-RECEIVED
3. ESTABLISHED	<-- <SEQ=300><ACK=101><CTL=SYN,ACK>	<-- SYN-RECEIVED
4. ESTABLISHED	--> <SEQ=101><ACK=301><CTL=ACK>	--> ESTABLISHED
5. ESTABLISHED	--> <SEQ=101><ACK=301><CTL=ACK><DATA>	--> ESTABLISHED

Basic 3-Way Handshake for Connection Synchronization

Figure 7.

Ex. 1008, §3.4.

98. Claim 1 recites: **1[b]** "determining an operating system identifier (OS ID) associated with the TCP session and the computing device;"

- a. I rely upon Deridder and/or Deridder in combination with the combined Tuvell-Lee malware system to disclose and render obvious this limitation, in combination with the knowledge of a POSITA.

b. I first note that this limitation only requires determining one OS ID.

Later claims, however, require determining multiple OS IDs. I therefore describe two ways in which this limitation is rendered obvious, either of which would be sufficient to meet the claim limitation.

c. **First OS ID** - Deridder discloses determining an OS ID associated with the TCP session and the computer on the other side of the NAT:

"An OS fingerprint uses IP and *TCP header fields from the initial TCP synchronization (SYN) packet of a connection to provide a unique signature or "fingerprint" of the OS* that generated the packet. The present invention makes use of most of the fields found in the passive OS fingerprinting tool 'p0f'. However, unlike 'p0f', the present system does not try to match the fingerprint to a list of known OS fingerprints, because it is concerned less with the characterization of computer systems and more with the detection and tracking of unique sessions. Therefore, the presence of multiple distinct fingerprints is enough to indicate multiple hosts behind a NAT device." Ex. 1006 at ¶ 31.

"Earlier proposed solutions have been based on operating system (OS) fingerprinting, Internet Protocol Identifier (IPID) analysis, and snooping network traffic headers and content. *Tools such as p0f (refer to: 'passive OS fingerprinting tool', Michal Zalewski, <lcantuf@coredump.cx>, 2000-2006; the entirety of which is hereby incorporated by reference) have been used in a non-real-time manner to gain information about distinct hosts behind a NAT.* These solutions do not address the problem of tracking multiple active hosts, behind a NAT device, simultaneously in real-time." Ex. 1006 at ¶ 4.

d. I was aware of p0f as of 2009. It was well-known in the computer networking industry as of that time. In 2009 when the Deridder application was filed, the operative version of p0f was 2.0.8, which was publicly available by September 6, 2006. Ex. 1010. The open-source

code from p0f provides further detail about the different protocol parameters used to determine an OS fingerprint, which I will call the first OS ID for reasons that will be explained in later claims.

- e. Deridder explains that it determines a particular OS Type, like

Windows, MacOS, and Linux:

"In an OS that provides a predictable IPID 610 such as Windows™ based operating system, ***When the OS type is determined to be of the Windows OS type***, a series of actions 612 to 622 are executed using algorithms directed to link-chain 612 and refresh detection 614, session activities at 616, IPID trendline at 618, and HTTP user agent 620 score as described above according to available criteria. In executing the method at 612 to 620, a weighted score is calculated at 622 from the available criteria and then compared with previously known weighted scores associated with other session entries to determine the session with the best matching score. When the weighted score exceeds a pre-determined threshold, YES at 518, the HTTP request is matched to a corresponding session at 520 and session data is updated at 522. When the weighted score does not exceed the pre-determined threshold, NO at 518, a new session is created at 530, provided that the HTTP request contains a sufficient number of the aforementioned criteria and appears to belong to a valid web browsing session. Sessions are not created for background processes such as, for example, software automatic updaters. ***When the OS type is determined to be an operating system that provides a predictable TCP timestamp at 640 such as a MacIntosh™/Linux OS type***, a TCP timestamp in the HTTP request is compared with a previously known TCP timestamps at 642. When the TCP timestamp matches a previously known TCP timestamp, YES at 518, the HTTP request is matched to a corresponding session at 520 and session data is update at 522. When the TCP timestamp does not match the previously known TCP timestamp, NO at 518, a new session is created 530. ***When the OS type is determined to be an operating system that does not provide either a predictable IPID or predictable TCP timestamp***, additional criteria may be used for determining a relevant session signature score." Ex. 1006 at ¶¶ 47-49.

f. In sum, Deridder describes using p0f to determine a first OS ID, which includes the type of OS.

g. **Second OS ID** - Tuvell discloses that its network analyzer/malware scanner stores OS ID information in the malware data store:

"In an exemplary embodiment, the malware data store 106 maintains information obtained based upon network traffic analysis, including, but not limited to, Internet protocol (IP) address of the network level packet analyzer and the time at which the packet was detected. The *malware data store 106 maintains records regarding the infected data, such as virus name, infected file name, infected file size, infected packet size and infected packet number*. The *malware data store 106 also maintains packet source related information, such as the source IP, source port and even source identifier (e.g., phone number)*. Moreover, destination information such as destination IP address, destination port and destination phone number can be recorded for analysis and reporting. The malware data store 106 can also maintain a record of the particular protocol name used for transmission of the packet." Ex. 1004 at ¶ 52.

"Infection reports 404 typically comprise information such as, but not limited to, detailed virus/threat vector information and mobile device related information, *including type of mobile device 308, operating system, software and versions*, and user information and mobile device 308 identifier. In an exemplary embodiment, the infection report 404 contains product identification that identifies the client malware scanner 310 software. For example, product identification includes, but is not limited to, a product identifier, major version identifier, minor version identifier and also a patch version as follows:

'productid+majorversion+minorversion+patchversion.' The infection report 404 can also include the infected filename and a unique identifier for the infected application, the name of the malware infection and the date and time of the infection. In addition, the infection report 404 can include mobile device 308 information, such as the identification of the mobile phone (e.g., phone number), firmware of the particular mobile device 308 (e.g., operating system information) and the software version of the mobile device 308." Ex. 1004 at ¶ 49.

"In certain embodiments, the analysis component 104 ***analyzes and correlates malware data obtained by the receiver component 102 and/or maintained by the malware data store 106.*** In particular, the analysis component 104 correlates data obtained from a variety of sources (e.g., network traffic, data network traffic and individual mobile devices 308). One function of CoreStats 100 is to assist mobile network administrators and operators to monitor threats to the mobile network 302 thereby identifying the mobile network's 302 vulnerability to malware. Early detection of the vulnerability helps them take better preventative measures. CoreStats 100 reports the spreading pattern of malware using collected information from individual mobile devices 308 as well as the network traffic. On the mobile network 302, malware can spread over using short range transmission protocols (e.g., Bluetooth, Infrared), long range or standard network protocols (e.g., TCP/IP, Messaging) or a combination of short and long range protocols. Hence, in order to facilitate reports of infections and spreading patterns of malware across the mobile network 302, CoreStats 100 uses information regarding the infections found in mobile devices 308 as well as those malware found in the network traffic by the network analyzer 202 and malware scanner 204. In particular, CoreStats 100 can generate spreading statistics of long range malware, such as malware that spreads using the mobile network 302 via TCP/IP, Messaging, and/or other protocols. Furthermore, CoreStats 100 can generate spreading statistics of short range malware, such as malware that spreads over Bluetooth, memory cards, or other means without being transported across the mobile network 302." Ex. 1006 at ¶ 54.

- h.** In my opinion, a POSITA would therefore have understood that, after the correlation, Tuvell's system will determine the operating system (OS ID) for a particular computing device associated with the network traffic or TCP session.
- i.** Furthermore, it is also my opinion that it would have been obvious to combine Tuvell with Lee for the reasons I already discussed above (Paragraphs 91-94). Lee discloses extracting the "operating system (OS)

of the aforesaid extracted node" from the network traffic. Ex. 1005 at ¶ 55. In my opinion, it would have been obvious to a POSITA that the User-Agent field in the HTTP header contains that information and could be easily determined by just looking at that field. Ex. 1007 at §§14.43, 3.8. It is my opinion, that, in reviewing Lee, POSITA would have known to use the User-Agent field in the HTTP header to determine the OS.

14.43 User-Agent

The User-Agent request-header field contains information about the user agent originating the request. This is for statistical purposes, the tracing of protocol violations, and automated recognition of user agents for the sake of tailoring responses to avoid particular user agent limitations. User agents SHOULD include this field with requests. The field can contain multiple product tokens (section 3.8) and comments identifying the agent and any subproducts which form a significant part of the user agent. By convention, the product tokens are listed in order of their significance for identifying the application.

User-Agent = "User-Agent" ":" 1*(product | comment)
 Example:

User-Agent: CERN-LineMode/2.15 libwww/2.17b3

3.8 Product Tokens

Product tokens are used to allow communicating applications to identify themselves by software name and version. Most fields using product tokens also allow sub-products which form a significant part of the application to be listed, separated by white space. By convention, the products are listed in order of their significance for identifying the application.

product = token ["/" product-version]
 product-version = token

Examples:

User-Agent: CERN-LineMode/2.15 libwww/2.17b3
 Server: Apache/0.8.4

Product tokens SHOULD be short and to the point. They MUST NOT be used for advertising or other non-essential information. Although any token character MAY appear in a product-version, this token SHOULD only be used for a version identifier (i.e., successive versions of the same product SHOULD only differ in the product-version portion of the product value).

j. Moreover, Tuvell already teaches the correlation of network traffic data with the operating system of a mobile device, as I described above (paragraph 82). See Ex. 1004 at ¶54. As such, when considering the teachings of Deridder, Tuvell, and Lee in combination, it would have

been obvious to determine an OS identifier associated with both the TCP session and the corresponding computer device. This identifier could be derived from various sources, including OS fingerprinting techniques and information found in the User-Agent field.

99. Claim 1 recites: **1[c]** "determining if malware is present in the TCP session and an associated malware ID by comparing a malware signature to the one or more TCP packets; and."

- a. I rely upon Tuvell to disclose and render obvious this limitation, in combination with the knowledge of a POSITA.
- b. Tuvell discloses determining if malware is present in the TCP session by comparing a malware signature against the TCP packets:

"In a further embodiment, the receiver component 102 includes a client data server 206 that receives reports of viruses or other malware from one or more client mobile devices. Individual mobile devices utilize scanning software to determine when malware is present and transmit malware or infection reports to the mobile network. ***Some representative malware scanning algorithms for mobile devices include, but are not limited to, malware signature searches; hash signature searches as described in U.S. patent application Ser. No. 11/697,647 'Malware Detection System and Method for Mobile Platforms';*** malware detection in headers and compressed parts of mobile messages as described in U.S. patent application Ser. No. 11/697,658 'Malware Detection System and Method for Compressed Data on Mobile Platforms'; malware modeling as described in U.S. patent application Ser. No. 11/697,642 'Malware Modeling Detection System and Method for Mobile Platforms'; malware modeling for limited access devices as described in U.S. patent application Ser. No. 11/697,664 'Malware Modeling Detection System and Method for Mobile Platforms'; and non-signature detection methods as described in U.S. patent application Ser. No. 11/697,668 'Non-Signature Malware Detection System and Method

for Mobile Platforms'. The malware reports include malware data, such as information regarding infected files, type, or name of infection. In an embodiment, the malware reports include device specific information such as current device hardware, software and/or an identifier for the infected mobile device (e.g., telephone number)." Ex. 1004 at ¶35.

c. Tuvell further discloses that there is a malware ID (virus name)

associated with the TCP connection:

"In an exemplary embodiment, the malware data store 106 maintains information obtained based upon network traffic analysis, including, but not limited to, Internet protocol (IP) address of the network level packet analyzer and the time at which the packet was detected. ***The malware data store 106 maintains records regarding the infected data, such as virus name, infected file name, infected file size, infected packet size and infected packet number.*** The malware data store 106 also maintains packet source related information, such as the source IP, source port and even source identifier (e.g., phone number). Moreover, destination information such as destination IP address, destination port and destination phone number can be recorded for analysis and reporting. The malware data store 106 can also maintain a record of the particular protocol name used for transmission of the packet." Ex. 1004 at ¶52.

d. Furthermore, it is my opinion that a POSITA would have understood that

a virus name must be linked to a detected malware signature—otherwise, it would not be possible to include the virus name in the record following a signature match.

100. Claim 1 recites: **1[d]** "generating an alert identifying a network address associated with the access device, the malware ID and the OS ID associated with TCP session that generated the alert."

- a. I rely upon Deridder in combination with Tuvell and Lee to disclose and render obvious this limitation, in combination with the knowledge of a POSITA.
- b. I already partially addressed this above in connection 1[b] (paragraph 98). Tuvell discloses generating an alert identifying a network address associated with the access device, malware ID, OS ID associated with the TCP session that generated the alert:

"At reference number 508 a determination is made as to whether to generate output, such as a report or alert. The determination can be based in whole or in part upon the malware data obtained from various sources. For example, if analysis indicates high levels of malware activity or significant impact on mobile network 302 performance, the determination is made to generate a report and alert or notify network administrators. Alternatively, reports are triggered periodically or upon operator request. In particular, operators can request particular reports via a user interface 108." Ex. 1004 at ¶ 60.

"Mitigating actions include preventative steps to avoid or inhibit spreading and/or effects of malware in the mobile network 302. In an embodiment, mitigating actions include update of a network analyzer and or malware scanner to capture and identify additional types of malware. In still other embodiments, a mitigation component 110 notifies a mobile device 308 user, force an update of mobile device 308 software, or even disable the mobile device's 308 data connections." Ex. 1004 at ¶ 62.

- c. In my opinion, a POSITA would have found it obvious to associate the OS types identified that I set forth in 1[b] (paragraph 98), as described in Deridder and Lee, with the malware identifier stored in the malware store 106, as taught by Tuvell. Tuvell, for instance, explains that the system's

analysis component 104 "correlates data obtained from a variety of sources (e.g., network traffic, data network traffic and individual mobile devices 308)." Ex. 1004 at ¶54. Given this disclosure, it is my opinion that it would have been apparent to a POSITA that such correlation would include OS type information determined through OS fingerprinting as disclosed in Deridder and via the User-Agent field as in Lee, particularly since Tuvell already contemplates utilizing data to determine OS type. In addition, it is my opinion that a POSITA would have recognized that the network address would naturally be linked to the access device—such as a router or NAT—especially considering that Deridder monitors traffic from the NAT on the service provider side.

- d. In my opinion, a POSITA would have found it obvious that including all available stored data—such as the router/NAT's network address, the malware identifier, and the operating system identifier—in the alert described in Tuvell would be a necessary. It would have been obvious to a POSITA that the alert system in Tuvell should incorporate enough detail to allow the network administrator to carry out the suggested mitigation actions, including the ability to "forc[e] an update of mobile device 308 software, or even disable the mobile device's 308 data connection." Ex. 1004, ¶ 62. Moreover, in my opinion, a POSITA would

recognize that providing the NAT's network address is essential for the network administrator to pinpoint the alert's origin within the network. Additionally, a POSITA would understand the importance of including the OS and malware IDs, as this information is critical for determining the appropriate software updates—since such updates are dependent on the specific operating system and may vary based on the malware involved.

5. Claim 2

101. In my opinion, each and every limitation of Claim 2 is rendered obvious by the combination of Deridder in view of Tuvell and Lee.

102. Claim 2 recites: **2[pre]** "The method of claim 1 wherein determining the operating system ID further comprises:"

- a. I incorporate my analysis from claim 1 herein. Combining the OS fingerprinting methods disclosed in Deridder with the Tuvell-Lee system—which already determines an OS ID based on network data packets—would have been obvious to a POSITA, for the reasons I previously discussed. In light of the limitations set forth in claim 2 and subsequent dependent claims, I employ the following mapping: (1) the OS ID obtained through Deridder's fingerprinting techniques corresponds to the "first OS ID," and (2) the OS ID derived from packet-based data

extraction, as made obvious by Lee, represents the "second OS ID." It is also my opinion that a POSITA would have recognized that the designations "first" and "second" in the claim could be used interchangeably. I explain this further below.

103. Claim 2 recites: **2[a]** "determining a protocol associated with the TCP packets and matching an OS fingerprint from one or more protocol parameters if present in the TCP packets to determine a first operating system identifier (ID);"

- a. I rely upon Deridder to disclose and render obvious this limitation, in combination with the knowledge of a POSITA.
- b. For the first OS ID, Deridder discloses determining a protocol associated with the TCP packets and matching an OS fingerprint from one or more protocol parameters if present in the TCP packets to determine an OS ID:

"An OS fingerprint uses IP and TCP header fields from the initial TCP synchronization (SYN) packet of a connection to provide a unique signature or "fingerprint" of the OS that generated the packet. The present invention makes use of most of the fields found in the passive OS fingerprinting tool 'p0f'. However, unlike 'p0f', the present system does not try to match the fingerprint to a list of known OS fingerprints, because it is concerned less with the characterization of computer systems and more with the detection and tracking of unique sessions. Therefore, the presence of multiple distinct fingerprints is enough to indicate multiple hosts behind a NAT device." Ex. 1006 at ¶ 31.

- c. In my opinion, a POSITA would recognize that when the inspection device in Deridder identifies the presence of IP fields within a TCP packet, it is effectively determining that the TCP packet is associated

with the IP protocol. Additionally, a POSITA would understand that the IP and TCP header fields qualify as protocol parameters, since they are specific to and defined by their respective protocols.

d. Specifically, Deridder discloses using the open-source software tool for passive OS fingerprinting called "p0f," which Deridder incorporates by reference in its entirety. Ex. 1004 at ¶¶4, 31. In my opinion, the source code from p0f, which is open source, meaning it was publicly available to anyone to download and read, provides further detail about the protocol parameters from TCP used to determine the OS fingerprint. *See* Section V.C, *supra*.

e. In sum, it is my opinion that Deridder discloses using p0f to determine a first OS ID that identifies the type of OS.

104. Claim 2 recites: **2[b]** "determining an application parameter associated with the TCP session if present, and identifying a second OS ID from the application parameter; and"

- a. I rely upon Tuvell and Lee to disclose and render obvious this limitation, in combination with the knowledge of a POSITA.
- b. I already explained in connection with 1[b] (paragraph 98) that it is my opinion that it would have been obvious to a POSITA that the Tuvell-Lee system makes it obvious to extract the OS ID by reading the HTTP User-

Agent field, as the HTTP specification explicitly states that the User-Agent may include information identifying the operating system.

- c. And, since HTTP operates at the application layer, it is my opinion that a POSITA would have understood that the User-Agent field constitutes an application parameter used to identify a second OS ID.

105. Claim 2 recites: **2[c]** "wherein the OS ID in the alert comprises one of the first OS ID or second OS ID associated with one or more computing devices coupled to the access device."

- a. I rely upon Deridder in combination with Tuvell and Lee to disclose and render obvious this limitation, in combination with the knowledge of a POSITA.

- b. I already explained in connection with 1[c] (paragraph 99) that Tuvell describes generating an alert that may involve "notify[ing] network administrators" or prompting further "mitigating actions," such as "forc[ing] an update of mobile device 308 software, or even disable the mobile device's 308 data connection." *See* Ex. 1004 at ¶¶60–62. In my opinion, a POSITA would have found it obvious that such alerts to the network administrator should include either the first or second OS ID (as determined using the methods disclosed in Deridder and Lee). For example, it is my opinion that it would be apparent to a POSITA that

knowing the OS type would be valuable for a network administrator to assess which operating systems might be susceptible to the detected malware. Furthermore, including the OS ID would also be necessary to enable software updates as described in Tuvell, since determining the appropriate update would require knowledge of the specific operating system installed on the device. *See* Ex. 1004 at ¶ 62:

"Mitigating actions include preventative steps to avoid or inhibit spreading and/or effects of malware in the mobile network 302. In an embodiment, mitigating actions include update of a network analyzer and or malware scanner to capture and identify additional types of malware. In still other embodiments, a mitigation component 110 notifies a mobile device 308 user, ***force an update of mobile device 308 software***, or even disable the mobile device's 308 data connections."

6. Claim 3

106. In my opinion, each and every limitation of Claim 3 is rendered obvious by the combination of Deridder in view of Tuvell and Lee.

107. Claim 3 recites: "The method of claim 2 further comprising sending a notification to a subscriber associated with the network address associated with the access device, the notification identifying a remediation portal to remove the determined malware in the alert."

a. I rely upon Tuvell to disclose and render obvious this limitation, in combination with the knowledge of a POSITA.

b. Tuvell teaches that, upon detecting malware, the system may notify a specific "mobile device 308 *user*," referring to the subscriber associated with the network address tied to the access device. Ex. 1004 at ¶62. In the context of Deridder's NAT-based architecture, a POSITA would have understood that the "mobile device user" corresponds to the user of the computer device behind the NAT. Deridder, in turn, discloses utilizing the TCP timestamp option to attempt to uniquely identify the mobile device associated with a given TCP session:

"For Mac® and Linux OS variants, the TCP timestamp option is used to estimate the uptime of the host. Although this value may be pseudo-random at boot time, it has been shown to increment in a highly predictable manner and *is used to distinguish multiple Mac and/or Linux clients behind a NAT device*. TCP timestamps can increment at different rates for different operating systems. This can be accommodated in the session matching algorithm by not requiring an exact match, but rather by allowing a timestamp to fall within a configurable range of an expected value." See Ex. 1006 at ¶37.

c. Based on these disclosures, it is my opinion that a POSITA would have found it obvious that the malware detection system should notify the *user* of the mobile device linked to the specific TCP session where the malware was identified.

d. Additionally, Tuvell also describes the system's ability to initiate a forced update of the mobile device software. See Ex. 1004 at ¶62. Tuvell explains that this functionality is managed by the "mitigation component

110 [i.e., remediation portal]," which Tuvell explains is responsible for initiating actions to mitigate or reduce the impact of malware within a mobile network. *Id.* at ¶ 30

"In a further embodiment, CoreStats 100 ***includes a mitigation component 110 that initiates and takes actions to mitigate or alleviate the impact of malware in a mobile network.*** The mitigation component 110 gathers information from the receiver component 102 as well as the malware analysis generated by the analysis component 104. In an embodiment, the mitigation component 110 uses this information to dynamically change the parameters of the scanning algorithms utilized to detect the presence of malware either in network traffic or on individual mobile devices and to modify the malware detection algorithms used to identify malware. Some representative malware algorithms include, but are not limited to, malware signature searches; hash signature searches as described in U.S. patent application Ser. No. 11/697,647 'Malware Detection System and Method for Mobile Platforms'; and malware detection in headers and compressed parts of mobile messages as described in U.S. patent application Ser. No. 11/697,658 'Malware Detection System and Method for Compressed Data on Mobile Platforms'." *Id.* at ¶ 30.

- e. In my opinion, a POSITA would therefore have found it obvious that, for the system to compel a user device to install an update, the notification would need to identify the mitigation component so that the device could retrieve and apply the appropriate software update.
- f. In my opinion, a POSITA also would have found it obvious that, in order to successfully force a software update, the remediation portal would need to ensure removal of the identified malware referenced in the alert. For instance, a POSITA would understand that updating the mobile device would involve downloading and installing updated software from

the mitigation component, and that this update process would, by its nature, eliminate the malware. It would have been clear to a POSITA that triggering an update that failed to remove the malware would serve no meaningful purpose, as the device would remain compromised if the infection persisted.

7. Claim 4

108. In my opinion, each and every limitation of Claim 4 is rendered obvious by the combination of Deridder in view of Tuvell and Lee.

109. Claim 4 recites: **4[pre]** "The method of claim 3 further comprising:"

a. I addressed claim 3 above and incorporate my analysis herein (paragraph 107).

110. Claim 4 recites: **4[a]** "identifying at the remediation portal the subscriber accessing the remediation portal;"

a. I rely upon Tuvell to disclose and render obvious this limitation, in combination with the knowledge of a POSITA.

b. As I explained for claim 3 (paragraph 107), Tuvell describes mitigation component 110 as functioning as a remediation portal capable of sending notifications to the "mobile device 308 *user*." See Ex. 1004 at ¶62. In the context of Deridder's NAT-based architecture, it is my opinion that a POSITA would have understood that the "mobile device user"

corresponds to the user of the computer device behind the NAT. Given that one of the mitigation component's functions is to deliver notifications to the user's device, it is my opinion that it would have been apparent to a POSITA that the component must necessarily identify the relevant user (or subscriber) in order to carry out this action.

111. Claim 4 recites: **4[b]** "identifying the OS associated with the computing device used by the subscriber to access the remediation portal;"

- a.** I rely upon Tuvell to disclose and render obvious this limitation, in combination with the knowledge of a POSITA.
- b.** As I explained for claim 3 (paragraph 107), as disclosed in Tuvell, one of the mitigation actions performed by the mitigation component is to force an update of the user's computer device. *See* Ex. 1004 at ¶62. In my opinion, a POSITA would have found it obvious that, in order to carry out such an update, the mitigation component would first need to verify the operating system associated with the user's device, to ensure compatibility and proper deployment of the update.

112. Claim 4 recites: **4[c]** "comparing the OS associated with the computing device to the at least one of the first OS ID or second OS ID identified in the alert; and"

- a. I rely upon Deridder in combination with Tuvell and Lee to disclose and render obvious this limitation, in combination with the knowledge of a POSITA.
- b. As I explained for 4[b] (paragraph 111), to update the device, the mitigation component necessarily identified the OS. In my opinion, it would have been obvious to a POSITA to compare the operating system information verified by the mitigation component with the first or second OS ID as determined in 2[a] and 2[b] by the methods in Deridder and Lee, which I explained above. It is my opinion that a POSITA would have understood the importance of performing such a comparison—particularly in scenarios where a user operates multiple devices that share the same network address. In such cases, matching the OS ID ensures that mitigation actions, like forced updates, are applied to the correct device. Accordingly, it is my opinion that a POSITA would have been motivated to validate the OS identified by the mitigation component against the OS data obtained as outlined in claim 2.

113. Claim 4 recites: **4[d]** "providing malware remediation to the computing device when the at least one of the first OS ID or second OS ID matches the OS associated with the computing device."

- a.** I rely upon Deridder in combination with Tuvell and Lee to disclose and render obvious this limitation, in combination with the knowledge of a POSITA.
- b.** In my opinion, a POSITA would have found it obvious that a forced update of a mobile device's software should only be performed if the operating system identified by the mitigation component matches the OS type determined as set forth in claim 2 by the methods in Deridder and Lee (paragraphs 102-105). In my opinion, a POSITA would understand that the malware would have been detected on a device running a particular OS—identified in claim 2—and that the update would only be necessary for devices running that specific OS. This is because it is well known that malware typically exploits vulnerabilities tied to particular applications on specific operating systems and may not be effective against devices running different OSs. Furthermore, a POSITA would appreciate that the update itself would likely be tailored to address the vulnerability on a particular OS, making it inapplicable to devices running other systems. Therefore, if a user were connecting with a device operating on a different OS, it would have been obvious that no update would be necessary for that device.

8. Claim 5

114. In my opinion, each and every limitation of Claim 5 is rendered obvious by the combination of Deridder in view of Tuvell and Lee.

115. Claim 5 recites: "The method of claim 3 wherein if the OS of the computing device is not the at least one of the first OS ID or second OS ID identified in the alert an indication is provided to the subscriber to connect to the remediation portal with an other computing device connected to the access device."

- a.** I rely upon Deridder in combination with Tuvell and Lee to disclose and render obvious this limitation, in combination with the knowledge of a POSITA.
- b.** Given that multiple devices may be connected to the NAT device at different times in the architecture described by Deridder, it is my opinion that a POSITA would have recognized that the specific device associated with the detected malware might not be connected to the NAT at the time the notification is to be delivered. In such a case, it is my opinion that it would have been obvious to a POSITA to send the notification to another device belonging to the same user, instructing the user to connect the affected device to the mitigation component in order to initiate the remediation process.

c. As I explained for 4[c] (paragraph 112), in my opinion, a POSITA would have understood that a user may own and operate multiple computing devices—each potentially running a different operating system—all of which connect to the network through the same network address. If the OS associated with a particular device did not match the OS determined by the methods in Deridder and Lee, a POSITA would have reasonably concluded that the infected device could be a different one belonging to the same user. As a result, a POSITA would have been motivated to notify the user that another one of her devices may be infected and that she should connect that device to the mitigation component. The motivation for such a notification is clear: it would have been obvious to a POSITA that the infected device should be remediated as quickly as possible to prevent potential harm to the device or its software.

9. Claim 6

116. In my opinion, each and every limitation of Claim 6 is rendered obvious by the combination of Deridder in view of Tuvell and Lee.

117. Claim 6 recites: "The method of claim 3 wherein the alert is cleared when all computing devices associated with the access device and the computing device having the at least one of the first OS ID or second OS ID have accessed the remediation portal."

- a. I rely upon Deridder in combination with Tuvell and Lee to disclose and render obvious this limitation, in combination with the knowledge of a POSITA.
- b. As I explained for claim 3 (paragraph 107), one possible mitigation action is to force an update of the user's mobile device. In my opinion, a POSITA would have found it obvious that any alert presented to the user should be cleared once the update has been completed, as it would be illogical to continue displaying an alert that no longer applies. Furthermore, a POSITA would understand that if the user has only one device running a particular OS, the alert would be cleared once all devices matching either the first or second OS ID as determined by the methods in Deridder and Lee have been updated.

10. Claim 7

118. In my opinion, each and every limitation of Claim 7 is rendered obvious by the combination of Deridder in view of Tuvell and Lee.

119. Claim 7 recites: "The method of claim 2 wherein determining the first OS ID is performed by comparing one or more protocol parameters of the one or more TCP packets having a SYN flag associated with the session, wherein the OS fingerprints identify the protocol parameters of the SYN flagged packet and the OS ID associated with parameters."

- a. I rely upon Deridder to disclose and render obvious this limitation, in combination with the knowledge of a POSITA.
 - b. As I discussed in 2[a] (paragraph 103), Deridder discloses generating an OS fingerprint by utilizing "IP and TCP header fields from the initial TCP synchronization (SYN) packet of a connection." Ex. 1006 at ¶31.
- In my opinion, a POSITA would have understood that this fingerprinting process involves comparing protocol parameters within packets marked with the SYN flag to identify the OS ID associated with those parameters. According to the TCP standard, the SYN flag is present in the first two packets of the three-way handshake. *See* Ex. 1008, §3.4.

3.4. Establishing a connection

The "three-way handshake" is the procedure used to establish a connection. This procedure normally is initiated by one TCP and responded to by another TCP. The procedure also works if two TCP simultaneously initiate the procedure. When simultaneous attempt occurs, each TCP receives a "SYN" segment which carries no acknowledgment after it has sent a "SYN". Of course, the arrival of an old duplicate "SYN" segment can potentially make it appear, to the recipient, that a simultaneous connection initiation is in progress. Proper use of "reset" segments can disambiguate these cases.

Several examples of connection initiation follow. Although these examples do not show connection synchronization using data-carrying segments, this is perfectly legitimate, so long as the receiving TCP doesn't deliver the data to the user until it is clear the data is valid (i.e., the data must be buffered at the receiver until the connection reaches the ESTABLISHED state). The three-way handshake reduces the possibility of false connections. It is the

implementation of a trade-off between memory and messages to provide information for this checking.

The simplest three-way handshake is shown in figure 7 below. The figures should be interpreted in the following way. Each line is numbered for reference purposes. Right arrows (→) indicate departure of a TCP segment from TCP A to TCP B, or arrival of a segment at B from A. Left arrows (←), indicate the reverse. Ellipsis (...) indicates a segment which is still in the network (delayed). An "XXX" indicates a segment which is lost or rejected. Comments appear in parentheses. TCP states represent the state AFTER the departure or arrival of the segment (whose contents are shown in the center of each line). Segment contents are shown in abbreviated form, with sequence number, control flags, and ACK field. Other fields such as window, addresses, lengths, and text have been left out in the interest of clarity.

TCP A		TCP B
1. CLOSED		LISTEN
2. SYN-SENT	→ <SEQ=100><CTL=SYN>	→ SYN-RECEIVED
3. ESTABLISHED	← <SEQ=300><ACK=101><CTL=SYN,ACK>	← SYN-RECEIVED
4. ESTABLISHED	→ <SEQ=101><ACK=301><CTL=ACK>	→ ESTABLISHED
5. ESTABLISHED	→ <SEQ=101><ACK=301><CTL=ACK><DATA>	→ ESTABLISHED

Basic 3-Way Handshake for Connection Synchronization

Figure 7.

- c. I have also analyzed the source code of p0f, which I understand is incorporated into Deridder by reference. In my opinion, the source code confirms that parameters from these two packets. *See* Section V.C, *supra*.

11. Claim 8

120. In my opinion, each and every limitation of Claim 8 is rendered obvious by the combination of Deridder in view of Tuvell and Lee.

121. Claim 8 recites: "The method of claim 2 wherein determining the application parameter associated with the session and identifying the second OS ID from the application parameter by inspecting an HTTP user agent string to determine the second OS ID."

- a. I rely upon Lee to disclose and render obvious this limitation, in combination with the knowledge of a POSITA.
- b. I already explained in 1[b] and 2[b] (paragraphs 98, 104), that it is my opinion that a POSITA would have found it obvious that the system described in Lee obtains the second OS ID by extracting it from the User-Agent field. This is because the HTTP specification explicitly indicates that OS information is included in the User-Agent field. Accordingly, a POSITA would have been motivated to examine the User-Agent field (or user agent string) to determine the second OS ID.

12. Claim 9

122. In my opinion, each and every limitation of Claim 9 is rendered obvious by the combination of Deridder in view of Tuvell and Lee.

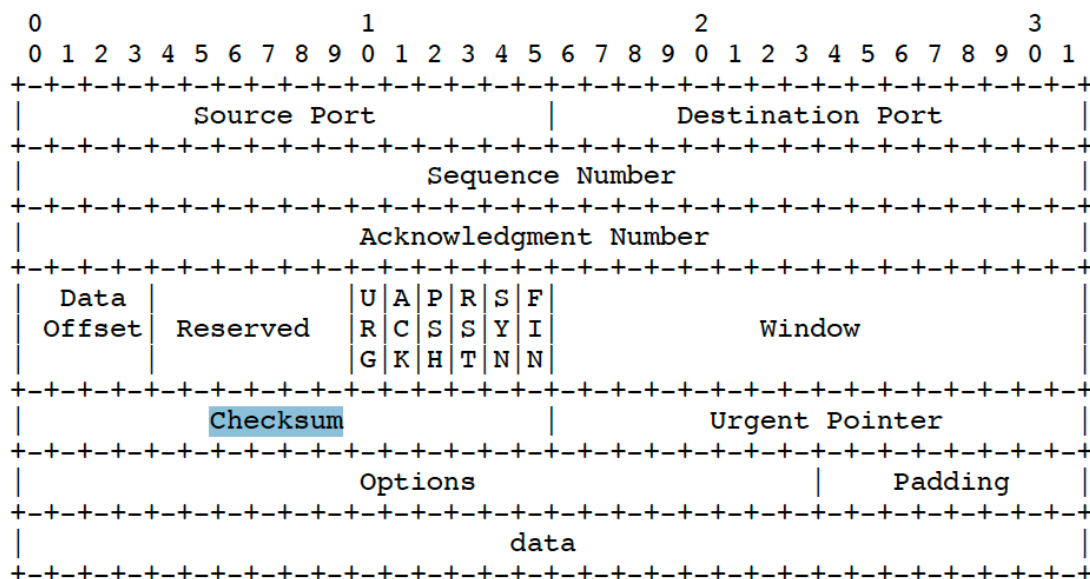
123. Claim 9 recites: "The method of claim 2 further comprising verifying the first OS ID against the second OS ID, wherein if the second OS ID does not match the first OS ID the session is identified by the OS ID identified as a more reliable OS ID."

- a.** I rely upon Deridder in combination with Tuvell and Lee to disclose and render obvious this limitation, in combination with the knowledge of a POSITA.
- b.** Because the combined teachings of Tuvell, Lee, and Deridder make it obvious to identify the OS ID through two distinct methods, it is my opinion that a POSITA would have found it obvious that discrepancies between these OS IDs could arise. For example, the OS fingerprinting based on TCP SYN packets might produce an incorrect OS ID. Alternatively, the OS ID found in the User-Agent field could be outdated or deliberately falsified as a security precaution. There is also the possibility that packet corruption might cause mismatches.
- c.** Consequently, in my opinion, it would have been obvious to verify the two OS IDs against each other as a form of error checking—a technique well known and commonly applied in computer science and networking. Indeed, the TCP standard itself includes a checksum mechanism to ensure packet reliability. See Ex. 1008, §3.1. It is therefore my opinion that a POSITA would have been motivated to perform such a comparison of OS IDs. In cases where the two OS IDs do not match, it would have been obvious to select the more reliable one for further use.

3.1. Header Format

TCP segments are sent as internet datagrams. The Internet Protocol header carries several information fields, including the source and destination host addresses [2]. A TCP header follows the internet header, supplying information specific to the TCP protocol. This division allows for the existence of host level protocols other than TCP.

TCP Header Format



TCP Header Format

Note that one tick mark represents one bit position.

Figure 3.

Checksum: 16 bits

The checksum field is the 16 bit one's complement of the one's complement sum of all 16 bit words in the header and text. If a segment contains an odd number of header and text octets to be checksummed, the last octet is padded on the right with zeros to form a 16 bit word for checksum purposes. The pad is not transmitted as part of the segment. While computing the checksum, the checksum field itself is replaced with zeros.

The checksum also covers a 96 bit pseudo header conceptually

13. Claim 10

124. In my opinion, each and every limitation of Claim 10 is rendered obvious by the combination of Deridder in view of Tuvell and Lee.

125. Claim 10 recites: "The method of claim 2 further comprising verifying the first OS ID against the second OS ID, wherein if the second OS ID matches the first OS ID either OS ID is used."

- a. I rely upon Deridder and Lee to disclose and render obvious this limitation, in combination with the knowledge of a POSITA.
- b. As I explained in claim 9 (paragraph 122-123), it is my opinion that a POSITA would have found it obvious to compare the first and second OS IDs (as determined by the methods in Deridder and Lee) as a means of error checking. Furthermore, in my opinion, it would have been obvious to a POSITA that if the two OS IDs matched, either one could be used.

14. Claim 11

126. In my opinion, each and every limitation of Claim 11 is rendered obvious by the combination of Deridder in view of Tuvell and Lee.

127. Claim 11 recites: "The method of claim 2 further comprising verifying the first OS ID against the second OS ID, wherein if the second OS ID matches the first OS ID but the second OS ID provides additional information, the second OS ID is used."

- a. I rely upon Tuvell and Lee to disclose and render obvious this limitation, in combination with the knowledge of a POSITA.
- b. As I explained in claim 9, it is my opinion that it would have been obvious to a POSITA to compare the first and second OS IDs (as determined by the methods in Deridder and Lee) for purposes of error checking. In the event that the two IDs did not match, a POSITA would have been motivated to determine if there was additional information contained in the User-Agent field that provides the second OS ID. Based on the HTTP specification, a POSITA would have known that the User-Agent field may also include details such as the type of browser being used. *See* Ex. 1007 at §§14.43, 3.8. In my opinion, it would have been obvious to a POSITA that, if the User-Agent field includes browser information alongside the OS ID, this additional context would enhance the reliability of the User-Agent data, making it the preferable source to use.

14.43 User-Agent

The `User-Agent` request-header field contains information about the user agent originating the request. This is for statistical purposes, the tracing of protocol violations, and automated recognition of user agents for the sake of tailoring responses to avoid particular user agent limitations. User agents SHOULD include this field with requests. The field can contain multiple product tokens (section 3.8) and comments identifying the agent and any subproducts which form a significant part of the user agent. By convention, the product tokens are listed in order of their significance for identifying the application.

User-Agent = "User-Agent" ":" 1*(product | comment)

Example:

User-Agent: CERN-LineMode/2.15 libwww/2.17b3

3.8 Product Tokens

Product tokens are used to allow communicating applications to identify themselves by software name and version. Most fields using product tokens also allow sub-products which form a significant part of the application to be listed, separated by white space. By convention, the products are listed in order of their significance for identifying the application.

```
product          = token [ "/" product-version ]
product-version = token
```

Examples:

```
User-Agent: CERN-LineMode/2.15 libwww/2.17b3
Server: Apache/0.8.4
```

Product tokens SHOULD be short and to the point. They MUST NOT be used for advertising or other non-essential information. Although any token character MAY appear in a product-version, this token SHOULD only be used for a version identifier (i.e., successive versions of the same product SHOULD only differ in the product-version portion of the product value).

See Ex. 1007 at §§14.43, 3.8.

15. Claim 12

128. In my opinion, each and every limitation of Claim 12 is rendered obvious by the combination of Deridder in view of Tuvell and Lee.

129. Claim 12 recites: "The method of claim 3 wherein sending the notification to the subscriber associated with the access device further comprises replacing the network address of the access device with a subscriber ID."

- a. I rely upon Deridder to disclose and render obvious this limitation, in combination with the knowledge of a POSITA.
- b. Deridder discloses detecting network traffic from the NAT on the service provider side, where all traffic originating from the NAT shares the same IP address corresponding to the access device. *See Ex. 1006, Fig. 1, ¶¶ 21-23.*

"The popularity of network address translation (NAT) devices such as home wireless routers has given rise to an increasing number of households with

multiple computing devices sharing a single network connection. *These devices make it difficult to distinguish the network traffic of multiple clients within a local area network, as their traffic appears to emanate from a single source Internet Protocol (IP) address.*" Ex. 1006 at ¶ 3.

- c. In my opinion, a POSITA would have understood that, in order to send a notification to a specific user, the generic IP address would need to be replaced with a more specific identifier unique to that user. Such an identifier could include, for example, a local IP address or a MAC address, which a POSITA would understand to be a subscriber ID.

16. Claim 13

130. In my opinion, each and every limitation of Claim 13 is rendered obvious by the combination of Deridder in view of Tuvell and Lee.

131. Claim 13 recites: "The method of claim 12 wherein sending the notification to the subscriber associated with the access device further comprises aggregating one or more detailed alerts having the subscriber ID into a single alert summary associated with the subscriber ID."

- a. I rely upon Tuvell to disclose and render obvious this limitation, in combination with the knowledge of a POSITA.
- b. Tuvell discloses sending the notification to the subscriber associated with the NAT further comprises aggregating one or more detailed alerts having the subscriber ID into a single alert summary associated with the subscriber ID:

"At reference number 508 a determination is made as to whether to generate output, such as a report or alert. The determination can be based in whole or in part upon the malware data obtained from various sources. For example, if analysis indicates high levels of malware activity or significant impact on mobile network 302 performance, the determination is made to generate a report and alert or notify network administrators. Alternatively, **reports are triggered periodically or upon operator request**. In particular, operators can request particular reports via a user interface 108." Ex. 1004 at ¶60.

"If the determination is made to generate output, one or more reports or alerts are generated at reference number 510. **Such reports can include information for presentation for an operator, stored for later use, or used in determining appropriate mitigation**. If no reports are to be generated, or after generation is complete, the process continues at reference number 512, where a determination is made as to whether to take action to mitigate the effects of malware on the mobile network. If no action is to be taken, the process terminates. If mitigating actions are to be taken, the process continues at reference number 514." Ex. 1004 at ¶ 61.

c. One such mitigation includes sending notifications to "a mobile device

308 **user**." *Id.* at ¶61. Accordingly, Tuvell contemplates that

notifications to the user can be generated periodically and/or accumulated

for later delivery. In my opinion, it would have been obvious to a

POSITA that all alerts for a user during a given period—or those stored

up—would be consolidated into a single alert summary for that user.

Consistent with my reasoning in 4[a], it would also have been obvious

that this report is associated with a user, particularly since Tuvell

discloses notifying the **user** of the mobile device. *See* Ex. 1004 at ¶62.

Moreover, I already explained that in the context of Deridder's NAT

architecture, a POSITA would have understood that the mobile device user corresponds to the computer device user.

17. Claim 14

132. In my opinion, each and every limitation of Claim 14 is rendered obvious by the combination of Deridder in view of Tuvell and Lee.

133. Claim 14 recites: "The method of claim 12 wherein the one or more computing devices are coupled to the access device through a local area network (LAN), the access device providing network address translation (NAT) or is coupled to NAT device to share the network address of the access device."

- a. I rely upon Deridder to disclose and render obvious this limitation, in combination with the knowledge of a POSITA.
- b. As illustrated in Figure 1, which I reproduced below, Deridder discloses a system where multiple computing devices—104, 106, and 108—are connected through a local area network to a NAT or router (referred to as the access device) that performs network address translation services.

Ex. 1006 at ¶¶3, 21-23.

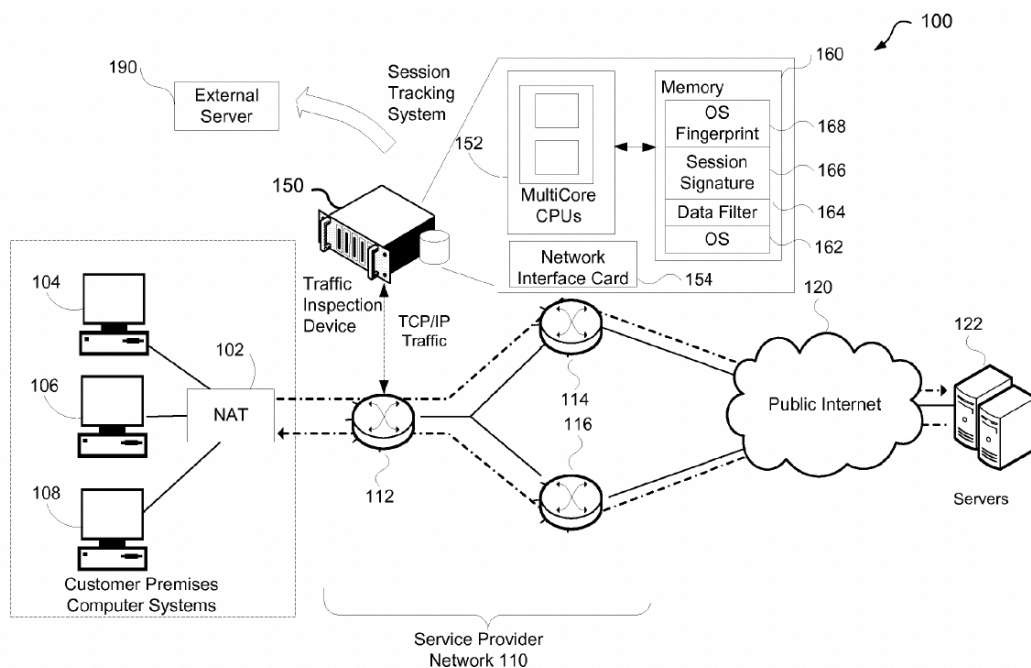


Figure 1

18. Claims 15 and 24

134. In my opinion, each and every limitation of Claims 15 and 24 is rendered obvious by the combination of Deridder in view of Tuvell and Lee.

135. I reviewed claims 15 and 24 and determined that they are substantially similar to claim 1, other than the preambles and preliminary limitations. I therefore only address the differences below

136. Claim 15 recites: **15[pre]** "A system for network based malware detection in a service provider network, the system comprising:"

- a. I rely upon Deridder in combination with Tuvell and Lee to disclose and render obvious this limitation, in combination with the knowledge of a POSITA.

b. I have been asked to assume that the preamble is limiting. I explained in 1[pre] (paragraph 96) that Deridder discloses that the traffic inspection device 150 is located within the service provider network 110, positioned outside of the NAT 102 that has the Tuvell-Lee malware system. Ex. 1006, Fig. 1, ¶¶21-23.

137. Claim 15 recites: **15[a]** "a plurality of network sensors coupled to the service provider network, each of the network sensors for:"

a. I rely upon Deridder in combination with Tuvell to disclose and render obvious this limitation, in combination with the knowledge of a POSITA.

b. In my opinion, a POSITA would have recognized that the functions of the traffic inspection device described in Deridder could be distributed across multiple sensors throughout the service provider network.

Deridder also contemplates various implementations of the traffic inspection device, including operation "inline with the network traffic."

Ex. 1006 at ¶21. In my opinion, a POSITA would have understood that operating inline could involve multiple network sensors. Moreover, Tuvell discloses an example of such an arrangement using a plurality of network sensors, and it would have been obvious to a POSITA that this

type of system could be employed within the service provider's network as described in Deridder. See Ex. 1004 at ¶¶73–93.

138. Claim 24 recites: **24[pre]** "A computer readable memory containing instructions for network based malware detection in a service provider network, the instructions when executed by a processor performing:"

- a. I rely upon Deridder in combination with Tuvell and Lee to disclose and render obvious this limitation, in combination with the knowledge of a POSITA.
- b. I have been asked to assume that the preamble is limiting. I explained in 1[pre] (paragraph 96) that Deridder discloses that the traffic inspection device 150 is located within the service provider network 110, positioned outside of the NAT 102 that has the Tuvell-Lee malware system. Ex. 1006, Fig. 1, ¶¶21-23. In my opinion, would have understood that the traffic inspection device must be implemented via computer-readable memory storing instructions that are executed by a processor.

139. Below, I provide a mapping as to why the other limitations of claims 15 and 24 are invalid for the reasons I discussed in connection with claim 1. I incorporate my analysis for claim 1 for claims 15 and 25 based upon this mapping.

Claim 15	Claim 24	Claim 1
15[b] receiving one or more transmission control protocol	24[a] receiving one or more transmission control protocol	1[a]

Claim 15	Claim 24	Claim 1
(TCP) packets originating from an access device coupled to the service provider network, the one or more TCP packets defining a TCP session between a computing device coupled to the access device, and a destination coupled to the service provider network;	(TCP) packets originating from an access device coupled to the service provider network, the one or more TCP packets defining a TCP session between a computing device coupled to the access device, and a destination coupled to the service provider network;	
15[c] determining an operating system identifier (OS ID) associated with the TCP session and the computing device associated;	24[b] determining an operating system identifier (OS ID) associated with the TCP session and the computing device;	1[b]
15[d] determining if malware is present in the TCP session and an associated malware ID by comparing a malware signature to the one or more TCP packets; and	24[c] determining if malware is present in the TCP session and an associated malware ID by comparing a malware signature to the one or more TCP packets; and	1[c]
15[e] generating an alert identifying a network address associated with the access device, the malware ID and the OS ID associated with TCP session that generated the alert.	24[d] generating an alert identifying a network address associated with the access device, the malware ID and the OS ID associated with TCP session that generated the alert.	1[d]

19. Claims 16-23, 25

140. In my opinion, each and every limitation of Claims 16-23 and 25 is rendered obvious by the combination of Deridder in view of Tuvell and Lee.

141. I have considered the dependent claims and determined that they are substantively the same as dependent claims I already addressed, and it is my opinion that they are invalid for the reasons set forth above. I provide the following mapping below and hereby incorporate my prior analysis for those claims herein.

Dependent Claim	Already Addressed Claim
16	2
17	3
18	12
19	7
20	8
21	9
22	10
23	11
25	4

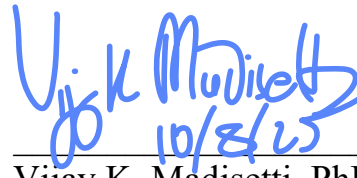
IX. CONCLUSION

142. As explained above, each and every element of the Challenged Claims of the '697 Patent is obvious to a POSITA in light of the above-discussed prior art references and is arranged or taught in those prior art references as recited in the claims.

143. I hereby declare 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 that such statements are made with the knowledge that willful false statements and the like are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Date: October 8, 2025

Respectfully submitted,



Vijay K. Madiseti, PhD

Appendix A

Dr. Vijay K. Madisetti

Fellow, IEEE

vkm@madisetti.com

Cell: 770-527-0177

Address:

225 Dodd Ln

Alpharetta, GA 30005

Employment:

- 1989-present: Full Professor of Cybersecurity & Privacy, and Electrical & Computer Engineering (**Georgia Tech, Atlanta, GA 30332**).

Areas of Technical Interest – Topics in Computer Science, Electrical Engineering, and Computer Engineering.

Education: **PhD** (EECS, University of California at Berkeley, 1989), **B.Tech** (Hons) in Electronics and Electrical Communications Engineering (Indian Institute of Technology, Kharagpur, 1984).

Startup Companies:

Director, **VP Technologies, Inc.** (1995-): A startup commercialized through Georgia Tech's Advanced Technology Development Corporation (ATDC) focusing on digital software and hardware design services for military market. <http://www.vptinc.net>

Director, **Soft Networks, LLC** (2001-2007): A startup commercialized through Georgia Tech support focusing on software development tools and compilers for Cellular/WiFi/VOIP/telecommunication products. <http://www.soft-networks.com>

Director, **Elastic Video Inc.** (2007- 2009): A startup commercialized through Georgia Tech's VentureLab (<http://venturelab.gatech.edu>) development image and video processing software for wireless & IP networking.

Litigation Experience (2020-2025) With Testimony

(Note: There may be multiple cases between the parties, e.g., District Court v. ITC, US versus Foreign Cases)

Masimo v. Apple

Case No. ITC 337-TA-1276 and related cases (Trade Secret & Delaware)

Expert for Masimo (2018-)

Technology: Pulse Oximetry

Submitted Expert Reports, Deposition & Trial

TQ Delta v. Nokia, Adtran, CommScope, 2Wire, et al

Cases in Delaware & Texas

Expert for TQ Delta (2019 -)

Technology: DSL and Cable Modems

Submitted Expert Reports, Depositions & Hearings.

Paltalk Holdings v. Cisco Systems Inc.

Western District of Texas (6:21-CV-00757-ADA)

Expert for Paltalk Holdings (2022-2024)

Technology: VOIP Technologies

Submitted Expert Reports and Deposition

Ollnova Technologies v. Carrier Group, Ecobee, et al

Southern District of Florida (Case 9:22-cv-80388-DMM) & ED Texas

Expert for Ollnova (2022-2023)

Technology: IoT

Submitted Report and Deposition & Trial

VideoLabs v. Asus, et al

Case No: ITC 337-TA-1323

Expert for VideoLabs (2022-2023)

Technology: Video Conferencing

Reports and Depositions

3GKL/KPN v. LG/Gemalto, TCL, HTC, Blackberry

Case No: 1:17-cv-00086-LPS

Expert for KPN/3GL (2017-2023)

Technology: Wireless Communications, Video Conferencing

Reports, Depositions, Trial

ViaTech v. Adobe

District of Delaware (CA No. 20-358-RGA-MPT)

Expert for ViaTech (2019-2023)
Technology: Software Licensing
Reports, Deposition, and Trial Testimony

Zilkr v. Cisco

ED Texas, Marshall (Case 2:22-CV-166-JRB-RSP)
Expert for Zilkr (2022-)
Technology: Webex
Reports & Deposition

HouseCanary versus Amrock/Intuit

District Court Bexar County, Tx (2016-CI-06300)
Expert for House Canary (2023-)
Technology: Machine Learning
Reports & Deposition

SpaceTime 3D v. Apple

WD Texas Waco (1-23-cv-00553-ADA)
Expert for SpaceTime 3D
Technology: User Interfaces (2022-)
Reports & Deposition

Pantech v. LG

ED Texas Texarkana (5:22-cv-00113)
Expert for LG (2023-2024)
Technology: Multitasking in Mobile Platforms
Reports & Deposition.

General Access v. Verizon, T-Mobile, et al

ED Texas
Expert for General Access (2023-2025)
Technology: 4G and 5G telecom standards
Reports, Depositions, and Trials

Additional matters include declarations supporting IPRs at the PTAB for Google (US Patent 8,601,154), On Semiconductor (US Patent 6,212,079), Ubisoft (US Patent 5,490,216), Lumens JDA (2017-2019), Broadcom (WiFi), Sony (US Patent 6,101,534), Kia, (US Patent 5,530,431), Qualcomm, Ericsson, Daimler Benz, VW, Estech, Amazon, Ring, Digital Ally, On Semiconductor, LoganTree, Unified Patents, BMC Software, Masimo, Sentius, BMW, Facebook, LG, PAWS, Google, Daimler, FedEx, Surge LLC, and Dolby.

Earned Degrees

- 1. B. Tech (Hons), Electronics & Electrical Comm. Engineering**
Indian Institute of Technology (IIT), Kharagpur, India
1984.
- 2. Ph.D., Electrical Engineering & Computer Sciences (EECS)**
University of California (UC), Berkeley. CA
1989.

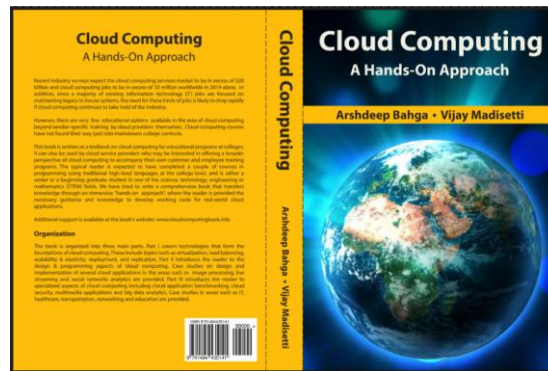
Books

- 1. VLSI Digital Signal Processors**
Madiseti, V.K.;
Boston: MA, IEEE Press: Butterworth Heinemann, 1995, 525 pp.
- 2. Quick-Turnaround ASIC Design in VHDL**
Romdhane, M., Madiseti, V.K., Hines, J.
Boston: MA, Kluwer Academic Press, 1996, 190 pp.
- 3. The Digital Signal Processing Handbook (First Edition)**
Madiseti, V. K., Williams, D. (Editors)
CRC Press, Boca Raton, Fla, 1998, 2500 pp.
- 4. VHDL: Electronics Systems Design Methodologies.**
Madiseti, V. K. (Editor)
Boston: MA, IEEE Standards Press, 2000, ISBN 0-7381-1878-8.
- 5. Platform-Centric Approach to System-on-Chip (SoC) Design.**
Madiseti, V. K., Arpnikanondt, A.
Springer, Boston: MA, Springer, 2004, 280 pp.
- 6. The Digital Signal Processing Handbook – Second Edition.**
Madiseti, V. K. (2009)
CRC Press, Boca Raton, Fla.
- 7. Cloud Computing: A Hands-On Approach**
A Bahga, V. Madiseti (2013)
KDP Press, 2013, 454 pp. Outside US Published by *Universities Press* (Ex-Pearson). Selected as *Best in Computing 2014* (By *ACM Computing Reviews*)

8. Internet of Things: A Hands-On Approach

A Bahga, V. Madiseti (2014)

KDP Press, 2014, 450 pp. Outside US Published by *Universities Press* (Ex-Pearson),

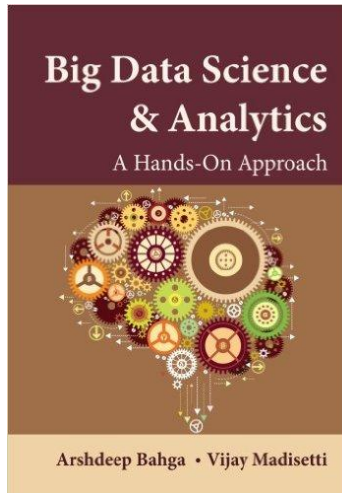


9. Big Data Science & Analytics: A Hands-On Approach

A Bahga, V. Madiseti (2016)

KDP Press, 2016, 542 pp.

[The book will tremendously benefit those who are interested in BDA: ACM Computing Reviews]

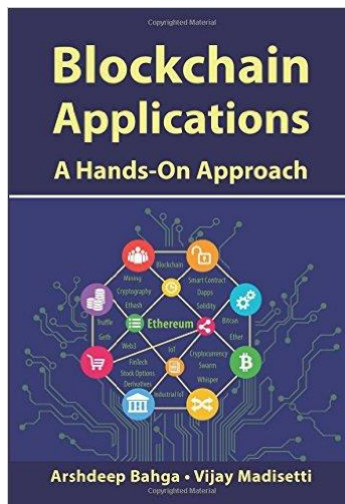


10. Blockchain Applications: A Hands-On Approach

A Bahga, V. Madiseti (2017)

KDP Press, 2017, 380 pp.

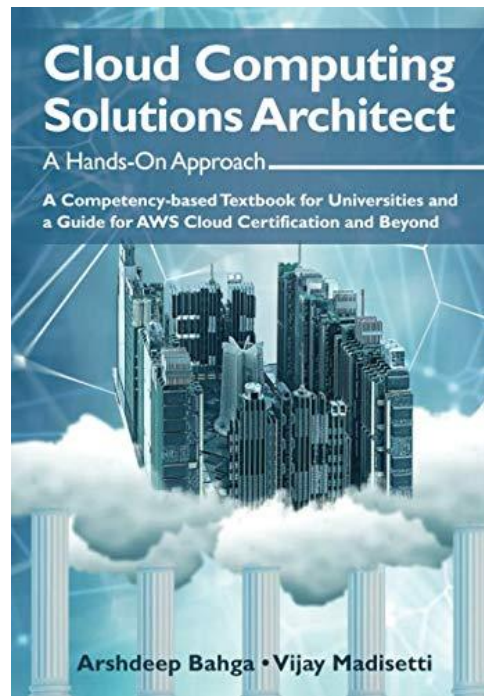
[The balance between explanation of architecture and more technical proof of concept through code is quite engaging: ACM Computing Reviews]



11. Cloud Solutions Architect: A Hands-On Approach

A Bahga, V. Madiseti (2020)

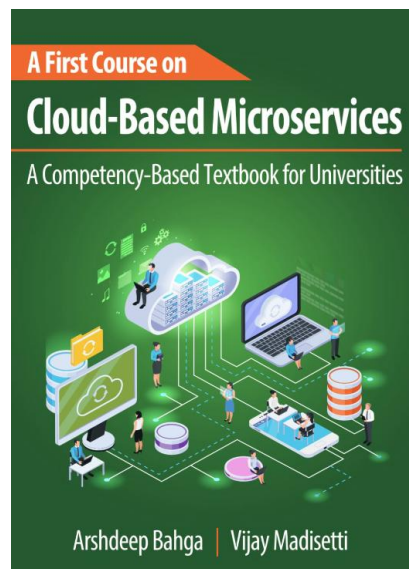
KDP Press, 2020, 930 pp.



12. A First Course in Cloud-Based Microservices

A Bahga, V. Madiseti (2024)

KDP Press, 2024, 590 pp.





Edited Books & Collection of Papers

- 1. Advances in Parallel & Distributed Simulation**
Madiseti, V.K.; Nicol, D., Fujimoto, R. (Editors)
San Diego, CA: SCS Press, 1991, 200 pp.
- 2. Modeling, Analysis, Simulation of Computer & Telecommunications Systems**
Madiseti, V., Gelenbe, E., Walrand, J. W. (Editors)
Los Alamitos: CA, IEEE Computer Society Press, 1994, 425 pp.
- 3. Modeling & Simulations on Microcomputers**
Madiseti, V.K. (Editor)
San Diego, CA: SCS Press, 138 pp. 1990.

Editorship of Journals & Transactions

- 1. IEEE Design & Test of Computers**
Special Issue: Reengineering Digital Systems
April – June 1999 (Vol 16, No 2)
Madiseti, V.K (Editor)
Los Alamitos: CA, IEEE Computer Society Press, 1999.

2. IEEE Design & Test of Computers

Special Issue: Rapid Prototyping of Digital Systems

Fall 1996 (Vol 13, No 3)

Madiseti, V., Richards, M. (Editors)

Los Alamitos: CA, IEEE Computer Society Press, 1994, 425 pp.

3. IEEE Transactions on Circuits & Systems II

Associate Editor: 1993-1995.

4. International Journal in Computer Simulation

Associate Editor: 1990-1993

5. International Journal in VLSI Signal Processing

Editorial Board: 1995 - Present

Issued US Patents (Recent)

- [1] 11,580,540 Methods and systems for smart contracts for security and filtering, Issued February 14, 2023
- [2] 11,556,928 Methods and systems for smart contracts for security and filtering, Issued January 17, 2023
- [3] 11,553,039 Service meshes and smart contracts for zero-trust systems, Issued January 10, 2023
- [4] 11,538,031 Method and system for identity and access management for blockchain interoperability, Issued December 27, 2022
- [5] 11,528,147 Verifying integrity and secure operations of cloud-based software services, Issued December 13, 2022
- [6] 11,526,879 Method and system for zero-knowledge and identity based key management for decentralized applications, Issued December 13, 2022
- [7] 11,494,764 Methods and systems for smart contracts for security and filtering, Issued November 8, 2022

- [8] 11,483,622 Hybrid blockchains and streamchains using non-crypto hashes for securing audio-, video- image-, and speech-based transactions and contracts, Issued October 25, 2022
- [9] 11,379,612 Method and system for securing cloud storage and databases from insider threats and optimizing performance, Issued July 5, 2022
- [10] 11,316,933 Service meshes and smart contracts for zero-trust systems, Issued April 26, 2022
- [11] 11,316,690 Blockchain token-based cloud orchestration architecture for discrete virtual network instances, Issued April 26, 2022
- [12] 11,314,695 Method and system for real-time collaboration and annotation-based action creation and management, Issued April 26, 2022
- [13] 11,283,865 Service meshes and smart contracts for zero-trust systems, Issued March 22, 2022
- [14] 11,250,394 Method and system for blockchain-based gemstone ownership, identity, custody, supply-chain transfer, trading, and secure payments, Issued February 15, 2022
- [15] 11,226,938 Method and system for real-time collaboration and event linking to documents, Issued January 18, 2022
- [16] 11,223,877 Hybrid blockchains and streamchains using non-crypto hashes for securing audio-, video-, image-, and speech-based transactions and contracts, Issued January 11, 2022
- [17] 11,222,331 Method and system for blockchain-based gemstone ownership, identity, custody, supply-chain transfer, trading, and secure payments, Issued January 11, 2022
- [18] 11,212,268 Method and system for identity and access management for blockchain interoperability, Issued December 28, 2021
- [19] 11,080,240 Method and system for real-time collaboration and annotation-based action creation and management, Issued August 3, 2021

- [20] 11,074,650 Method and system for blockchain-based gemstone ownership, identity, custody, supply-chain transfer, trading, and secure payments, Issued July 27, 2021
- [21] 10,983,768 Method and system for persistent helpers for functions as a service (FAAS) in cloud computing environments, Issued April 20, 2021
- [22] 10,853,772 Method and system for exchange of value or tokens between blockchain networks, Issued December 1, 2020
- [23] 10,579,643 Method and system for tuning blockchain scalability, decentralization, and security for fast and low-cost payment and transaction processing, Issued March 3, 2020
- [24] 10,503,927 Method and system for securing cloud storage and databases from insider threats and optimizing performance, Issued December 10, 2019
- [25] 10,460,283 Smart contract optimization for multiparty service or product ordering system, Issued October 29, 2019
- [26] 10,459,946 Method and system for tuning blockchain scalability, decentralization, and security for fast and low-cost payment and transaction processing, Issued October 29, 2019
- [27] 10,402,589 Method and system for securing cloud storage and databases from insider threats and optimizing performance, Issued September 3, 2019
- [28] 10,394,845 Method and system for tuning blockchain scalability for fast and low-cost payment and transaction processing, Issued August 27, 2019
- [29] 10,289,631 Method and system for tuning blockchain scalability for fast and low-cost payment and transaction processing, Issued May 24, 2019
- [30] 10,255,342 Method and system for tuning blockchain scalability, decentralization, and security for fast and low-cost payment and transaction processing, Issued April 9, 2019

- [31] 10,243,743 Tokens or crypto currency using smart contracts and blockchains, Issued March 26, 2019
- [32] 10,204,339 Method and system for blockchain-based combined identity, ownership, integrity and custody management, Issued February 12, 2019
- [33] 10,204,148 Method and system for tuning blockchain scalability, decentralization, and security for fast and low-cost payment and transaction processing, Issued February 12, 2019
- [34] 10,121,143 Method and system for blockchain-based combined identity, ownership, integrity and custody management, Issued November 6, 2018
- [35] 10,102,526 Method and system for blockchain-based combined identity, ownership, integrity and custody management, Issued October 16, 2018
- [36] 10,102,265 Method and system for tuning blockchain scalability for fast and low-cost payment and transaction processing, Issued October 16, 2018
- [37] 9,935,772 Methods and systems for operating secure digital management aware applications, Issued April 3, 2018
- [38] 9,769,213 Method and system for secure digital object

8 International Journal of Electronic Commerce Studies

Ranked #8 in the list of issued US patents in the area of blockchain

Ranking in the area of US patents issued in the area of blockchain technology in 2020

Table 3. Patent by assignee

Applicant (Assignee)	Patents	%	Applicant (Assignee)	Patents	%
IBM	47	14.0%	Wells Fargo Bank	4	1.2%
Accenture Global Solutions	28	8.4%	Alibaba	3	0.9%
Bank of America	24	7.2%	Dell	3	0.9%
Capital One Services	12	3.6%	The Bank Of New York Mellon	3	0.9%
Mastercard	11	3.3%	The Toronto-Dominion Bank	3	0.9%
Guardtime IP Holdings Limited	8	2.4%	Verizon Patent And Licensing	3	0.9%
Madiseti Vijay K.	8	2.4%	Winklevoss I.P.	3	0.9%
Black Gold Coin, Inc.	7	2.1%	SAP	2	0.6%
Cisco	7	2.1%	Sony	2	0.6%
Acronis International	5	1.5%	Stampery	2	0.6%
Blockchain Asics	5	1.5%	Uipco	2	0.6%
Tran Bao	5	1.5%	Vmware	2	0.6%
Intel	4	1.2%	Walmart Apollo	2	0.6%
Northern Trust	4	1.2%	Wiig Rex	2	0.6%
Shocard	4	1.2%	Xage Security	2	0.6%

management, Issued September 19, 2017

Refereed Journal Publications

1. Trends in the Electronic Control of Mine Hoists

*Madiseti, V. and Ramlu, M.,
IEEE Transactions on Industry Applications, Vol IA-22, No. 6,
November/December 1986. Pages 1105-1112*

2. Multilevel range/NEXT performance in digital subscriber loops

Brand, G.; Madiseti, V.; Messerschmitt, D.G.;
Communications, Speech and Vision, IEE Proceedings I [see also IEE
Proceedings-Communications] ,Volume: 136 , Issue: 2 , April 1989
Pages:169 – 174

3. Seismic migration algorithms on parallel computers

Madiseti, V.K.; Messerschmitt, D.G.;
Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and
Signal Processing, IEEE Transactions on], Volume: 39 , Issue: 7 , July 1991
Pages:1642 – 1654

4. Asynchronous algorithms for the parallel simulation of event-driven dynamical systems

Madiseti, V.K.; Walrand, J.C.; Messerschmitt, D.G.:
ACM Transactions on Modeling and Computer Simulation, v 1, n 3, July 1991,
Pages: 244-74

5. Synchronization mechanisms for distributed event-driven computation

Madiseti, V.K.; Hardaker, D.:
ACM Transactions on Modeling and Computer Simulation, v 2, n 1, Jan. 1992,
Pages: 12-51

6. Efficient VLSI Architectures for the Arithmetic Fourier Transform (AFT)

Kelley, B.T.; Madiseti, V.K.;
Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and
Signal Processing, IEEE Transactions on] ,Volume: 41 , Issue: 1 , January
1993
Pages:365-378

7. The fast discrete Radon transform. I. Theory

Kelley, B.T.; Madiseti, V.K.;
Image Processing, IEEE Transactions on ,Volume: 2 , Issue: 3 , July 1993
Pages:382 – 400

8. The Georgia Tech digital signal multiprocessor

Barnwell, T.P., III; Madisetti, V.K.; McGrath, S.J.A.;
Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and
Signal Processing, IEEE Transactions on] ,Volume: 41 , Issue: 7 , July 1993
Pages:2471 – 2487

9. The MIMDIX Environment for Parallel Simulation

Madisetti, V.K.; Hardaker, D.; Fujimoto, R.M.:
Journal of Parallel and Distributed Computing, v18, no. 4, August 1993,
Pages: 473-83.

10.LMSGEN: a prototyping environment for programmable adaptive digital filters in VLSI

Romdhane, M.S.B.; Madisetti, V.K.;
Chapter in VLSI Signal Processing, VII, 1994.,
Pages:33 – 42

11. Fixed-point co-design in DSP

Egolf, T.W.; Famorzadeh, S.; Madisetti, V.K.;
Chapter in VLSI Signal Processing, VII, 1994.,
Pages:113 - 126

12. A fast spotlight-mode synthetic aperture radar imaging system

Madisetti, V.K.;
Communications, IEEE Transactions on ,Volume: 42 , Issue: 234 , February-
April 1994
Pages:873 – 876

13. Rapid prototyping on the Georgia Tech digital signal multiprocessor

Curtis, B.A.; Madisetti, V.K.;
Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and
Signal Processing, IEEE Transactions on] ,Volume: 42 , Issue: 3 , March
1994
Pages:649 – 662

14.Low-power signaling in asymmetric noisy channels via spectral shaping

Sipitca, M.; Madisetti, V.K.;
Signal Processing Letters, IEEE, Volume: 1 , Issue: 8 , Aug 1994
Pages:117 – 118

15. A quantitative methodology for rapid prototyping and high-level synthesis of signal processing algorithms

Madisetti, V.K.; Curtis, B.A.;
Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and

Signal Processing, IEEE Transactions on] ,Volume: 42 , Issue: 11 , Nov. 1994
Pages:3188 – 3208

16. Computer Simulation of Application-Specific Signal Processing Systems

Casinovi, G.; Madisetti, V.K.;

International Journal in Computer Simulation, Vol. 4, No. 4, Nov 1994.

17. System partitioning of MCMs for low power

Khan, S.A.; Madisetti, V.K.;

Design & Test of Computers, IEEE ,Volume: 12 , Issue: 1 , Spring 1995
Pages:41 – 52

18. Error correcting run-length limited codes for magnetic recording

Jaejin Lee; Madisetti, V.K.;

Magnetics, IEEE Transactions on ,Volume: 31 , Issue: 6 , Nov. 1995
Pages:3084 – 3086

19. Virtual prototyping of embedded microcontroller-based DSP systems

Madisetti, V.K.; Egolf, T.W.;

Micro, IEEE ,Volume: 15 , Issue: 5 , Oct. 1995
Pages:9 – 21

20. Constrained multitrack RLL codes for the storage channel

Lee, J.; Madisetti, V.K.;

Magnetics, IEEE Transactions on ,Volume: 31 , Issue: 3 , May 1995
Pages:2355 – 2364

21. Rapid digital system prototyping: current practice, future challenges

Madisetti, V.K.;

Design & Test of Computers, IEEE ,Volume: 13 , Issue: 3 , Fall 1996
Pages:12 – 22

22. Conceptual prototyping of scalable embedded DSP systems

Dung, L.-R.; Madisetti, V.K.;

Design & Test of Computers, IEEE ,Volume: 13 , Issue: 3 , Fall 1996
Pages:54 – 65

23. Advances in rapid prototyping of digital systems

Madisetti, V.K.; Richards, M.A.;

Design & Test of Computers, IEEE ,Volume: 13 , Issue: 3 , Fall 1996
Pages:9

24. Combined modulation and error correction codes for storage channels

Jaemin Lee; Madiseti, V.K.;

Magnetics, IEEE Transactions on ,Volume: 32 , Issue: 2 , March 1996

Pages:509 – 514

25. Model-based architectural design and verification of scalable embedded DSP systems-a RASSP approach

Dung, L.-R.; Madiseti, V.K.; Hines, J.W.;

Chapter in VLSI Signal Processing, IX, 1996.

Pages:147 – 156

26. Low-power digital filter implementations using ternary coefficients

Hezar, R.; Madiseti, V.K.;

Chapter in VLSI Signal Processing, IX, 1996.,

Pages:179 – 188

27. All-digital oversampled front-end sensors

Romdhane, M.S.B.; Madiseti, V.K.;

Signal Processing Letters, IEEE, Volume: 3 , Issue: 2 , Feb. 1996

Pages:38 – 39

28. Modeling COTS components in VHDL

Calhoun, S., Reese, R; Egolf, T., Madiseti, V.K.;

Journal of VLSI Signal Processing, Volume: 14 , Issue: 2 , Nov 1996

Pages: 24 – 31

29. VHDL-Based Rapid Systems Prototyping

Egolf, T.; Madiseti, V.K.;

Journal of VLSI Signal Processing, Volume: 14 , Issue: 2 , Nov 1996

Pages: 40-52

30. Interface design for core-based systems

Madiseti, V.K.; Lan Shen;

Design & Test of Computers, IEEE ,Volume: 14 , Issue: 4 , Oct.-Dec. 1997

Pages:42 - 51

31. Incorporating cost modeling in embedded-system design

Debardelaben, J.A.; Madiseti, V.K.; Gadiant, A.J.;

Design & Test of Computers, IEEE ,Volume: 14 , Issue: 3 , July-Sept. 1997

Pages:24 – 35

32. On homomorphic deconvolution of bandpass signals

Marenco, A.L.; Madiseti, V.K.;

Signal Processing, IEEE Transactions on [see also Acoustics, Speech, and

Signal Processing, IEEE Transactions on] ,Volume: 45 , Issue: 10 , Oct. 1997
Pages:2499 – 2514

33. A case study in the development of multi-media educational material: the VHDL interactive tutorial

Gadient, A.J.; Stinson, J.A., Jr.; Taylor, T.C.; Aylor, J.H.; Klenke, R.H.; Salinas, M.H.; Madiseti, V.K.; Egolf, T.; Famorzadeh, S.; Karns, L.N.; Carter, H.W.;
Education, IEEE Transactions on ,Volume: 40 , Issue: 4 , Nov. 1997
Pages:17 pp.

34. Adaptive mobility management in wireless networks

Jeongwook Kim; Madiseti, V.K.;
Electronics Letters ,Volume: 34 , Issue: 15 , 23 July 1998
Pages:1453 – 1455

35. Efficient implementation of two-band PR-QMF filterbanks

Hezar, R.; Madiseti, V.K.;
Signal Processing Letters, IEEE ,Volume: 5 , Issue: 4 , April 1998
Pages:92 – 94

36. On fast algorithms for computing the inverse modified discrete cosine transform

Yun-Hui Fan; Madiseti, V.K.; Mersereau, R.M.;
Signal Processing Letters, IEEE ,Volume: 6 , Issue: 3 , March 1999
Pages:61 – 64

37. System on chip or system on package?

Tummala, R.R.; Madiseti, V.K.;
Design & Test of Computers, IEEE ,Volume: 16 , Issue: 2 , April-June 1999
Pages:48 – 56

38. Reengineering legacy embedded systems

Madiseti, V.K.; Jung, Y.-K.; Khan, M.H.; Kim, J.; Finnessy, T.;
Design & Test of Computers, IEEE ,Volume: 16 , Issue: 2 , April-June 1999
Pages:38 – 47

39. Reengineering digital systems

Madiseti, V.K.;
Design & Test of Computers, IEEE ,Volume: 16 , Issue: 2 , April-June 1999
Pages:15 – 16

40. Parameter optimization of robust low-bit-rate video coders

Sangyoun Lee; Madiseti, V.K.;
Circuits and Systems for Video Technology, IEEE Transactions on, Volume: 9

Issue: 6 , Sept. 1999
Pages:849 – 855

41. Closed-form for infinite sum in bandlimited CDMA

Jatunov, L.A.; Madiseti, V.K.;
Communications Letters, IEEE ,Volume: 8 , Issue: 3 , March 2004
Pages:138 – 140

42. A new protocol to enhance path reliability and load balancing in mobile ad hoc networks

Argyriou, A.; Madiseti, V.K.;
Journal of Ad Hoc Networks, Elsevier Press, 2004

43. Closed-form analysis of CDMA systems using Nyquist pulse

Jatunov, L.A.; Madiseti, V. K.;
Communications Letters, IEEE (Under Revision), 2005.

44. Systematic Design of End-to-End Wireless Mobility Management Protocols,

Argyriou, A.; Madiseti, V. K.;
ACM/Springer Wireless Networks (WINET), Accepted 2005.

45. A Novel End-to-End Approach for Video Streaming Over the Internet,

Argyriou, A.; Madiseti, V. K.;
Kluwer Telecommunications Systems, Vol. 28, No. 2, Pages 133-150, Jan 2005. *Special Issue on Multimedia Streaming.*

46. An Analytical Framework of RD Optimized Video Streaming with TCP,

Argyriou, A.; Madiseti, V. K.;
IEEE Transactions on Multimedia, Submitted for review in March 2005.

47. Modeling the Effect of Handoffs on Transport Protocol Performance,

Argyriou, A.; Madiseti, V. K.;
IEEE Transactions on Mobile Computing, Submitted for review in March 2005

48. Throughput Models for Transport Protocols with CBR and VBR Traffic Workloads",

Argyriou, A.; Madiseti, V. K.;
ACM Transactions on Multimedia Computing, Communications & Applications, Submitted for review in April 2005.

49. "Electronic System, Platform & Package Codesign",

Madiseti, V. K.

IEEE Design & Test of Computers, Volume 23, Issue 3, June 2006. pages 220-233.

- 50. "The Design of an End-to-End Handoff Management Protocol",**
A. Argyriou, Madiseti, V. K.
Wireless Networks, Springer, May 2006.
- 51. "A Soft-Handoff Transport Protocol for Media Flows in Heterogeneous Mobile Networks ",**
A. Argyriou, Madiseti, V. K.
Computer Networks, Vol 50, Issue 11, Pages 1860-1871, August 2006.
- 52. "Computationally Efficient SNR Estimation for Bandlimited WCDMA Systems"**
L. Jatunov, Madiseti, V. K.
IEEE Transactions on Wireless Communications, Volume 5, Issue 13, December 2006, Pages 3480-3491.
- 53. "Space-Time Codes for Wireless & Mobile Applications",**
M. Sinnokrot, Madiseti, V.K.
DSP Handbook, Second Edition, 2009 (to be published)
- 54.A. Bahga, V. Madiseti, "Rapid Prototyping of Advanced Cloud-Based Systems",** *IEEE Computer*, vol. 46, no. 11, Nov 2013, pp 76-83, 2013
- 55.A. Bahga, V. Madiseti, "Cloud-Based Information Integration & Informatics Framework for Healthcare Applications",** *IEEE Computer*, February 2015.
- 56.A. Bahga, V. Madiseti, "A Cloud-based Approach for Interoperable EHRs",** *IEEE Journal of Biomedical and Health Informatics*, vol. 17, no. 5, Sep 2013, pp. 894-906, 2013
- 57.A. Bahga, V. Madiseti, "Cloud-Based Information Technology Framework for Data Driven Intelligent Transportation Systems",** *Journal of Transportation Technologies*, vol.3 no.2, April 2013
- 58.A. Bahga, V. Madiseti, "Performance Evaluation Approach for Multi-tier Cloud Applications",** *Journal of Software Engineering and Applications*, vol. 6, no. 2, pp. 74-83, Mar 2013.
- 59.Yusuf, A., V. Madiseti, "Configuration for Predicting Travel Time Using Wavelet Packets and Support Vector Regression",** *Journal of Transportation Technologies*, vol 3, no. 3, June 2013.

60. A. Bahga, V. Madiseti, "**Analyzing Massive Machine Maintenance Data in a Computing Cloud**", *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 10, pp. 1831 - 1843, 2012.
61. N. Radia, Y. Zhang, M. Tatimapula, V. Madiseti, "**Next Generation Applications on Cellular Networks: Trends, Challenges, and Solutions**," *Proceedings of the IEEE*, Vol 100, Issue 4, pp. 841-854, 2012.
62. A. Bahga, V. Madiseti, "**Rapid Prototyping of Advanced Cloud-Based Systems**", *IEEE Computer*, vol. 46, no. 11, Nov 2013, pp 76-83, 2013
63. A. Bahga, V. Madiseti, "**A Cloud-based Approach for Interoperable EHRs**", *IEEE Journal of Biomedical and Health Informatics*, vol. 17, no. 5, Sep 2013, pp. 894-906, 2013
64. A. Bahga, V. Madiseti, "**Cloud-Based Information Integration & Informatics Framework for Healthcare Applications**", *IEEE Computer*, February 2015.
65. A. Bahga, V. Madiseti, "**A Blockchain Platform for Industrial Internet of Things**", *Journal of Software Engineering and Applications*, Vol 9, No. 10, October 2016.
66. A. Bahga, V. Madiseti, R. Madiseti, and A. Dugenske, "**Software Defined Things for Manufacturing Networks**", *Journal of Software Engineering and Applications*, Vol 9, No. 9, September 2016.
67. A. Bahga, V. Madiseti, J. Corporan, "**Result-as-a-Service (RAAS): Persistent Helper Functions in a Serverless Offering**", *Journal of Software Engineering and Applications*, Vol 13, No 10, October 2020.
68. A. Bahga, V. Madiseti, "**A Value-Token Transfer Protocol (VTP) for Decentralized Finance**", *Journal of Software Engineering and Applications*, Vol 13, No. 11, February 2020
69. V. Madiseti, S. Panda, "**A Dynamic Leader Election Algorithm for Decentralized Networks**", *Journal of Transportation Technologies*, Vol 11, No 3, July 2021
70. M. Mohapatra, A. Phukan, V. K. Madiseti, "**Generative Adversarial Network Based Approach Towards Synthetically Generating Insider Threat Scenarios**", *Journal of Software Engineering*, Vol 16, No. 11, November 2023.

71. Z. Wase, V. K. Madiseti, A. Bahga, "**Object Detection Meets LLMs: Model Fusion for Safety and Security**", *Journal of Software Engineering*, Vol 16, No 12, December 2023
72. T. Singh, H. Aditya, V. Madiseti, A. Bahga, "**Whispered Tuning: Data Privacy Preservation In Fine-Tuning LLMs through Differential Privacy**", *Journal of Software Engineering*, Vol 17, No. 1, January 2024.
73. G. Dhingra, S. Chawla, V. K. Madiseti, A. Bahga, "**Smaller & Smarter: Score-Driven Network Chaining of Smaller Language Models**", *Journal of Software Engineering*, Vol 17, No. 1, January 2024
74. P. Rai, S. Sood, V. K. Madiseti, A. Bahga, "**GUARDIAN: A Multi-Tiered Defense Architecture for Thwarting Prompt Injection Attacks on LLMs**", *Journal of Software Engineering*, Vol 17, No 1, January 2024.
75. A. Shankar, V. Madiseti, **A Framework for Cybersecurity Alert Distribution and Response Network (ADRIAN)**, *Journal of Software Engineering and Applications*, Vol 17, No. 5, 2024.
76. X. Li, Vijay K. Madiseti, **ERAD: Enhanced Ransomware Attack Defense System for Healthcare Organizations**, *Journal of Software Engineering and Applications*, Vol 17, No. 5, 2024.
77. P. Kumar, Vijay K. Madiseti, **Sher: A Secure Broker for DevSecOps and CI/CD Workflows**, *Journal of Software Engineering and Applications*, Vol 17, No. 5, 2024.
78. M. Jaikanth, Vijay K. Madiseti, **A Comparative Analysis of Cybersecurity Threat Taxonomies for Healthcare Organizations**, *Journal of Software Engineering and Applications*, Vol 17, No. 5, 2024.
79. Y. Tan, Vijay K. Madiseti, **User Profile & Attitude Analysis Based on Unstructured Social Media and Online Activity**, *Journal of Software Engineering and Applications*, Vol 17, No. 6, 2024.
80. K. Bhatnagar, Vijay Madiseti, **Certis: Cloud Asset Management & Threat Evaluation Using Behavioral Fingerprinting at Application Layer**, *Journal of Software Engineering and Applications*, Vol 17, No. 6, 2024.
81. I. Edlabadkar, Vijay K. Madiseti, **Cybersecurity Risk Management through Behavior-Based Contextual Analysis of Online Logs**, *Journal of Software Engineering and Applications*, Vol 17, No. 6, 2024.

82. J. Delugio, Vijay K. Madiseti, **Enhanced Memory-Safe Linux Security Modules (eLSMs) for Improving Security of Docker Containers for Data Centers**, Journal of Software Engineering and Applications, Vol 17, No. 5, 2024.
 83. A. Biju, V. Ramesh, Vijay K. Madiseti, **Security Vulnerability Analyses of Large Language Models (LLMs) through Extension of the Common Vulnerability Scoring System (CVSS) Framework**, Journal of Software Engineering and Applications, Vol 17, No. 5, 2024.
 84. S. Madikeri, Vijay K. Madiseti, **Ad Blockers & Online Privacy: A Comparative Analysis of Privacy Enhancing Technologies (PET)**, Journal of Software Engineering and Applications, Vol 17, No. 5, 2024.
 85. M. Manikandawamy, Vijay K. Madiseti, **Design & Test of an Advanced Web Security Analysis Tool (AWSAT)**, Journal of Software Engineering and Applications, Vol 17, No. 5, 2024.
 86. H. Aditya, Vijay Madiseti, et al, **Evaluating Privacy Leakage and Memorization Attacks on Large Language Models (LLMs) in Generative AI Applications**, Journal of Software Engineering and Applications, Vol 17, No. 5, 2024.
-

Peer Reviewed Conference Publications

1. **Dynamically-reduced complexity implementation of echo cancelers**
Madiseti, V.; Messerschmitt, D.; Nordstrom, N.;
Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '86., Volume: 11, Apr 1986
2. **Seismic migration algorithms using the FFT approach on the NCUBE multiprocessor**
Madiseti, V.K.; Messerschmitt, D.G.;
Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988
International Conference on, 11-14 April 1988
3. **Seismic migration algorithms on multiprocessors**
Madiseti, V.K.; Messerschmitt, D.G.;
Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988
International Conference on, 11-14 April 1988
Pages:2124 - 2127 vol.4

- 4. WOLF: A rollback algorithm for optimistic distributed simulation systems**
Madiseti, V.; Walrand, J.; Messerschmitt, D.;
Simulation Conference Proceedings, 1988 Winter, December 12-14, 1988
Pages:296 – 305
- 5. Efficient distributed simulation**
Madiseti, V.; Walrand, J.; Messerschmitt, D.;
Simulation Symposium, 1989. The 22nd Annual, March 28-31, 1989
Pages:5 - 6
- 6. High speed migration of multidimensional seismic data**
Kelley, B.; Madiseti, V.;
Acoustics, Speech, and Signal Processing, 1991. ICASSP-91., 1991
International Conference on, 14-17 April 1991
Pages:1117 - 1120 vol.2
- 7. Performance of a fast analog VLSI implementation of the DFT**
Buchanan, B.; Madiseti, V.; Brooke, M.;
Circuits and Systems, 1992., Proceedings of the 35th Midwest Symposium
on, 9-12 Aug. 1992
Pages:1353 - 1356 vol.2
- 8. Task scheduling in the Georgia Tech digital signal multiprocessor**
Curtis, B.A.; Madiseti, V.K.;
Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE
International Conference on, Volume: 5, 23-26 March 1992
Pages:589 - 592 vol.5
- 9. The fast discrete Radon transform**
Kelley, B.T.; Madiseti, V.K.;
Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE
International Conference on, Volume: 3, 23-26 March 1992
Pages:409 - 412 vol.3
- 10. Yield-based system partitioning strategies for MCM and ASEM design**
Khan, S.; Madiseti, V.;
Multi-Chip Module Conference, 1994. MCMC-94, Proceedings., 1994 IEEE, 15-
17 March 1994
Pages:144 – 149
- 11. Multitrack RLL codes for the storage channel with immunity to intertrack interference**
Lee, J.; Madiseti, V.K.;

Global Telecommunications Conference, 1994. GLOBECOM '94.
'Communications: The Global Bridge', IEEE, Volume: 3 , 28 Nov.-2 Dec. 1994
Pages:1477 - 1481 vol.3

12. A parallel mapping of backpropagation algorithm for mesh signal processor

Khan, S.A.; Madiseti, V.K.;
Neural Networks for Signal Processing [1995] V. Proceedings of the 1995
IEEE Workshop, 31 Aug.-2 Sept. 1995
Pages:561 – 570

13. Virtual prototyping of embedded DSP systems

Madiseti, V.K.; Egolf, T.; Famorzadeh, S.; Dung, L.-R.;
Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995
International Conference on, Volume: 4 , 9-12 May 1995
Pages:2711 - 2714 vol.4

14. Assessing and improving current practice in the design of application-specific signal processors

Shaw, G.A.; Anderson, J.C.; Madiseti, V.K.;
Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995
International Conference on ,Volume: 4 , 9-12 May 1995
Pages:2707 - 2710 vol.4

15. Introduction to ARPA's RASSP initiative and education/facilitation program

Corley, J.H.; Madiseti, V.K.; Richards, M.A.;
Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995
International Conference on, Volume: 4, 9-12 May 1995
Pages:2695 - 2698 vol.4

16. DSP design education at Georgia Tech

Madiseti, V.K.; McClellan, J.H.; Barnwell, T.P., III;
Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995
International Conference on, Volume: 5 , 9-12 May 1995
Pages:2869 - 2872 vol.5

17. Rapid prototyping of DSP systems via system interface module generation

Famorzadeh, S.; Madiseti, V.K.;
Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference
Proceedings., 1996 IEEE International Conference on, Volume: 2 , 7-10 May
1996
Pages:1256 - 1259 vol. 2

18. Rapid prototyping of DSP chip-sets via functional reuse

Romdhane, M.S.B.; Madiseti, V.K.;

Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on ,Volume: 2 , 7-10 May 1996

Pages:1236 - 1239 vol. 2

19. A constructive deconvolution procedure of bandpass signals by homomorphic analysis

Marenco, A.L.; Madiseti, V.K.;

Geoscience and Remote Sensing Symposium, 1996. IGARSS '96. 'Remote Sensing for a Sustainable Future.', International, Volume: 3, 27-31 May 1996

Pages:1592 - 1596 vol.3

20. BEEHIVE: an adaptive, distributed, embedded signal processing environment

Famorzadeh, S.; Madiseti, V.; Egolf, T.; Nguyen, T.;

Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on, Volume: 1, 21-24 April 1997

Pages:663 - 666 vol.1

21. Target detection from coregistered visual-thermal-range images

Perez-Jacome, J.E.; Madiseti, V.K.;

Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on, Volume: 4, 21-24 April 1997

Pages:2741 - 2744 vol.4

22. Variable block size adaptive lapped transform-based image coding

Klausutis, T.J.; Madiseti, V.K.;

Image Processing, 1997. Proceedings., International Conference on ,Volume: 3 , 26-29 Oct. 1997

Pages:686 - 689 vol.3

23. A Rate 8/10 (0, 6) MTR Code And Its Encoder/decoder

Jaejin Lee; Madiseti, V.K.;

Magnetics Conference, 1997. Digests of INTERMAG '97., 1997 IEEE International, 1-4 April 1997

Pages:BS-15 - BS-15

24. VHDL models supporting a system-level design process: a RASSP approach

DeBardelaben, J.A.; Madiseti, V.K.; Gadiant, A.J.;

VHDL International Users' Forum, 1997. Proceedings, 19-22 Oct. 1997

Pages:183 - 188

25. A performance modeling framework applied to real time infrared search and track processing

Pauer, E.K.; Pettigrew, M.N.; Myers, C.S.; Madiseti, V.K.;
VHDL International Users' Forum, 1997. Proceedings, 19-22 Oct. 1997
Pages:33 – 42

26. System design and re-engineering through virtual prototyping: a temporal model-based approach

Khan, M.H.; Madiseti, V.K.;
Signals, Systems & Computers, 1998. Conference Record of the Thirty-Second Asilomar Conference on, Volume: 2 , 1-4 Nov. 1998
Pages:1720 - 1724 vol.2

27. Web-Enabled Information Appliances for Broadband Residential Networks”, F. Ahmed, V. Madiseti, Y. Jiao, V. Dasigi, Yamacraw IAB

Presentation, 2000. Atlanta, GA

28.Web-Enabled Transcoding for Broadband Residential Networks (BRAN), M. Rahim, V. Madiseti, K. Naik, V. Dasigi, Yamacraw IAB

Presentation, 2000, Atlanta, GA.

29. Enforcing Interdependencies and Executing Transactions Atomically Over Autonomous Mobile Data Stores Using SyD Link Technology, S.

K. Prasad, V. Madiseti, et al, IEEE International Conference on Distributed Computing, 2003.

30. A debugger RTOS for embedded systems

Akgul, T.; Kuacharoen, P.; Mooney, V.J.; Madiseti, V.K.;
Euromicro Conference, 2001. Proceedings. 27th, 4-6 Sept. 2001
Pages:264 - 269

31. Adaptability, extensibility and flexibility in real-time operating systems

Kuacharoen, P.; Akgul, T.; Mooney, V.J.; Madiseti, V.K.;
Digital Systems, Design, 2001. Proceedings. Euromicro Symposium on, 4-6 Sept. 2001
Pages:400 – 405

32. Effect of handoff delay on the system performance of TDMA cellular systems

Turkboydari, M.; Madiseti, V.K.;
Mobile and Wireless Communications Network, 2002. 4th International Workshop on, 9-11 Sept. 2002
Pages:411 – 415

33. Enforcing interdependencies and executing transactions atomically over autonomous mobile data stores using SyD link technology

Prasad, S.K.; Bourgeois, A.G.; Dogdu, E.; Sunderraman, R.; Yi Pan; Navathe, S.; Madiseti, V.;

Distributed Computing Systems Workshops, 2003. Proceedings. 23rd
International Conference on, 19-22 May 2003
Pages:803 – 809

34. Performance evaluation and optimization of SCTP in wireless ad-hoc networks

Argyriou, A.; Madiseti, V.;
Local Computer Networks, 2003. LCN '03. Proceedings. 28th Annual IEEE
International Conference on, 20-24 Oct. 2003
Pages:317 - 318

35. Implementation of a calendar application based on SyD coordination links

Prasad, S.K.; Bourgeois, A.G.; Dogdu, E.; Sunderraman, R.; Yi Pan; Navathe, S.; Madiseti, V.;
Parallel and Distributed Processing Symposium, 2003. Proceedings.
International, 22-26 April 2003
Pages:8 pp.

36. Bandwidth aggregation with SCTP

Argyriou, A.; Madiseti, V.;
Global Telecommunications Conference, 2003. GLOBECOM '03. IEEE Volume:
7, 1-5 Dec. 2003
Pages:3716 - 3721 vol.7

37. Software streaming via block streaming

Kuacharoen, P.; Mooney, V.J.; Madiseti, V.K.;
Design, Automation and Test in Europe Conference and Exhibition, 2003
Pages:912 – 917

38. Frequency-dependent space-interleaving for MIMO OFDM systems

Mohajerani, P.; Madiseti, V.K.;
Radio and Wireless Conference, 2003. RAWCON '03. Proceedings, Aug. 10-13, 2003
Pages:79 - 82

39. A media streaming protocol for heterogeneous wireless networks

Argyriou, A.; Madiseti, V.;
Computer Communications, 2003. CCW 2003. Proceedings. 2003 IEEE 18th
Annual Workshop on, 20-21 Oct. 2003
Pages:30 – 33

40. Realizing load-balancing in ad-hoc networks with a transport layer protocol

Argyriou, A.; Madiseti, V.;
Wireless Communications and Networking Conference, 2004. WCNC. 2004

IEEE, Volume: 3, 21-25 March 2004
Pages:1897 - 1902 Vol.3

41.Streaming H.264/AVC video over the Internet

Argyriou, A.; Madisetti, V.;
Consumer Communications and Networking Conference, 2004. CCNC 2004.
First IEEE, 5-8 Jan. 2004
Pages:169 – 174

Other Publications

- 1. A Transport Layer Technology for Improving the QoS of Networked Multimedia Applications <draft-madisetti-arguriou-qos-sctp-00.txt>.**
Madisetti, V., Argyriou, A.
IETF Internet-Draft, Jul 25, 2002.
- 2. Voice & Video over Mobile IP Networks <draft-madisetti-argyriou-voice-video-mip-00.txt>**
Madisetti, V., Argyriou, A.
IETF Internet-Draft, Nov 20, 2002.
- 3. Enhancements to ECRTTP with Applications to Robust Header Compression for Wireless Applications. <draft-madisetti-rao-suresh-rohc-00.txt>**
Madisetti, V.; Rao, S., Suresh, N.
IETF Internet-Draft, June 30, 2003.

Ph.D. Students Graduated

- 1. Brian T. Kelley, 1992**
VLSI Computing Architectures for High Speed Signal Processing
Member of Technical Staff, Motorola

Winner of Dr. Thurgood Marshall Dissertation Fellowship Award
- 2. Bryce A. Curtis, 1992**
Special Instruction Set Multiple Chip Computer for DSP
Member of Technical Staff, IBM
- 3. Jaejin Lee, 1994**
Robust Multitrack Codes for the Magnetic Channel
Professor, Yonsei University, Korea

- 4. Mohamed S. Ben Romdhane, 1995**
Design Synthesis of Application-Specific IC for DSP
Director of IP, Rockwell
- 5. Shoab A. Khan, 1995**
Logic and Algorithm Partitioning on MCMs
Professor, National University of Science & Technology, Pakistan
- 6. Lan-Rong Dung, 1997**
VHDL-based Conceptual Prototyping of Embedded DSP Architectures
Professor, National Chaio Tung University, Taiwan

Winner of VHDL International Best PhD Thesis Award, 1997
- 7. Thomas W. Egolf, 1997**
Virtual Prototyping of Embedded DSP Systems
Distinguished Member of Technical Staff, Agere
- 8. Alvaro Marenco, 1997**
On Homomorphic Deconvolution of Bandpass Signals
Professor, Texas A&M University

Winner of GIT ECE Outstanding Teaching Assistant Award
- 9. Shahram Famorzadeh, 1997**
BEEHIVE: A Distributed Environment for Adaptive Signal Processing
Member of Technical Staff, Rockwell
- 10. Timothy J. Klausutis, 1997**
Adaptive Lapped Transforms with Applications to Image Coding.
US Air Force/Univ. of Florida
- 11. Lan Shen, 1998**
Temporal Design of Core-Based Systems
Member of Technical Staff, IBM
- 12. James DeBardelaben, 1998**
Optimization Based Approach to Cost Effective DSP Design
Research Scientist, Johns Hopkins University

Georgia Tech ECE Faculty Award
- 13. Sangyoun Lee, 1999**
Design of Robust Video Signal Processors
Professor, Yonsei University

US Army Sensors Lab Research Excellence Award, 1999
- 14. Rahmi Hezar, 2000**
Oversampled Digital Filters
Member of Technical Staff, Texas Instruments
- 15. Yong-kyu Jung, 2001**

Model-Based Processor Synthesis
Professor, Texas A&M University

16. Mustafa Turkboylari, 2002

Handoff Algorithms for Wireless Applications
Member of Technical Staff, Texas Instruments

17. Yun-Hui Fan, 2002

A Stereo Audio Coder with Nearly Constant Signal to Noise Ratio
Post-Doctoral Research Associate, Northeastern University

18. Subrato K. De, 2002

Design of a Retargetable Compiler for DSP
Member of Technical Staff, Qualcomm

US Army Sensors Lab Research Excellence Award, 1999

19. Chonlameth Aripnikanondt, 2004

System-on-Chip Design with UML
Professor, King Mongkut's University, Thailand

US Army Sensors Lab Research Excellence Award, 1999

20. Loran Jatunov, 2004

Performance Analysis of 3G CDMA Systems
Senior Research Scientist, Soft Networks, LLC. Now at Samsung

21. Antonios Argyriou, 2005, Research Scientist, Philips

22. Pilho Kim, 2009, Scientist, VP Technologies, Inc.

23. M. Sinnokrot, 2009, Staff Engineer, Apple

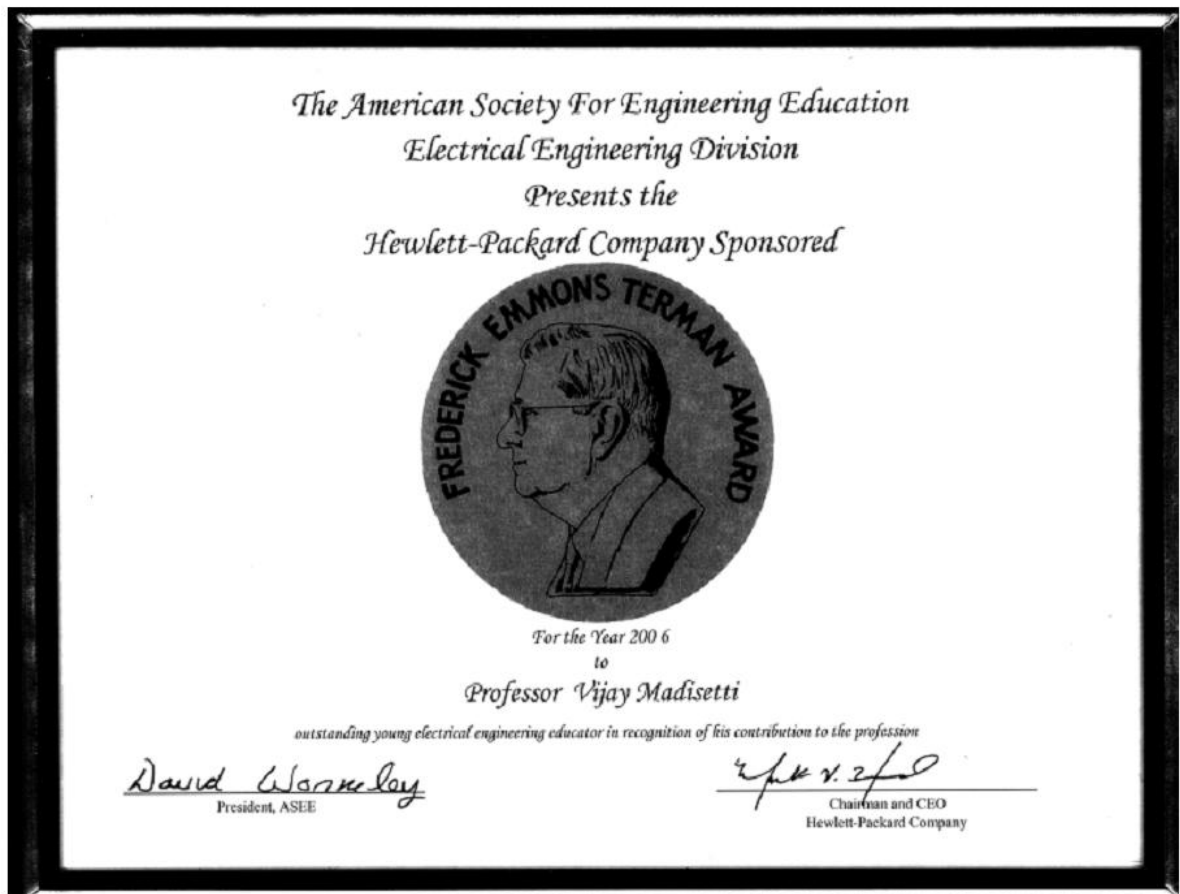
24. A. Yusuf, 2013, Professor, National University, Pakistan

25. Joel Corporan, 2023, Senior Scientist, Warner Media, USA

Awards & Honors

1. **Jagasdis Bose National Science Talent Fellowship**, Indian Institute of Technology, Kharagpur, 1980-1984
2. **General Proficiency Prize**, Indian Institute of Technology, Kharagpur, 1984
3. **Demetri Angelakos Outstanding Graduate Student Award**, Univ. of California, Berkeley, 1989

4. **Ira Kay IEEE/ACM Best Paper Award** for Best Paper presented at IEEE Annual Simulation Symposium, 1989
5. **IBM Faculty Development Award** 1990
6. **Technical Program Chair**, IEEE Workshop on Parallel and Distributed Simulation. 1990
7. **Technical Program Chair**, IEEE MASCOTS'94
8. **NSF RI Award**, 1990
9. **VHDL International Best PhD Dissertation Advisor Award**, 1997
10. **Georgia Tech Outstanding Doctoral Dissertation Advisor Award**, 2001
11. **ASEE 2006 Frederick Emmons Terman Medal**, 2006
12. **Fellow of IEEE**



Intellectual Property Disclosures (Georgia Tech)

<u>Patent</u>	<u>Date</u>	<u>Description</u>
2843	2004	Method and Apparatus for Improving the Performance of Wireless LANs
2825	2003	Method and Apparatus for Optimal Partitioning and Ordering of Antennas for Layered Space-Time Block Codes in MIMO Communications Systems
2815	2003	How to Rapidly Develop a SyD Application
GSU-023	2003	Rapid Development of SyD Applications
2810	2003	System on Mobile Devices Middleware Design
2718	2003	A Transport Layer Algorithm for Improved Anycast Communication
2717	2003	A Novel Transport Layer Load-Balancing Algorithm
2716	2003	A Transport Layer QoS Algorithm
2715	2003	A Novel Transport Layer Algorithm for MPLS Performance
2659	2002	A New Algorithm and Technology for Implementing Mobile IP with Applications to Voice and Video over Mobile IP
2656	2002	Debugging with Instruction-Level Reverse Execution
2655	2002	Embedded Software Streaming
2539		System of Databases: An Enabling Technology for Programming
2517	2002	A Dynamic Instantiated Real-Time Operating System Debugger
2516	2002	A Dynamic Real-Time Operating System
GSU-009	2001	System of Databases: Architecture,, Global Queries, Triggers and Constraints
2480	2001	Mobile Fleet Application based on SyD Technology
2479	2001	System of Databases: A model with coordination links and a calendar application
1893	1999	Beehive
1726	1995	Very High Scale Integrated Circuit Hardware Description Language Models (VHDL Models)
1401	1995	Self-Compensation Receiver (SCR)