UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

CROWDSTRIKE, INC.,
Petitioner

v.

GOSECURE, INC.,
Patent Owner

Case No. IPR2025-00070
Patent No. 9,954,872

DECLARATION OF DR. MARKUS JAKOBSSON IN SUPPORT OF PETITION FOR *INTER PARTES* REVIEW OF U.S. PATENT NO. 9,954,872

TABLE OF CONTENTS

I. INTRODUCTION AND QUALIFICATIONS 16
A. Educational Background and Professional Experience 16
II. METHODOLOGY; MATERIALS CONSIDERED 25
III. OVERVIEW AND LEGAL STANDARDS 27
A. Person of Ordinary Skill in the Art 27
B. Obviousness 28
C. Analogous Art 33
D. Claim Construction 34
IV. LEVEL OF A PERSON OF ORDINARY SKILL 35
V. OVERVIEW OF THE TECHNOLOGY 36
A. Intrusion Detection Systems 36
B. Honeypots/Decoy Systems 40
C. Virtual Machines, Hypervisors/Virtual Machine Monitors 43
D. Fingerprints/Signatures 47
E. Intrusion Detection System Hardware 48
VI. OVERVIEW OF THE '872 PATENT 52
VII. THE PROSECUTION HISTORY OF THE '872 PATENT 58
A. Prosecution History of the '872 Patent 58
VIII. SUMMARY OF UNPATENTABILITY 59
IX. OVERVIEW OF THE PRIOR ART 61
A. Overview of *Capalik* 61
B. Overview of *Goel* 66
C. Overview of *King* 68
D. Overview of *Pike* 72
E. Overview of *Farley* 74
X. Ground 1: Capalik renders Claims 1-6, 12, 15, 20-21 Obvious 77
A. Claim [1(Pre)]: "A computer implemented method of identifying unauthorized activities on a first computer

system attached to a computer network, wherein the first computer system comprises one or more processors and memory, the method comprising:" 77

B. Claim [1(a)]: "monitoring activity on the first computer system;" 83

1. *'872 Patent's* VM and VMM 84
2. *Capalik's* VM and VMM 86

C. Claim [1(b)]: "identifying a plurality of activities being performed at the first computer system, wherein each of the activities includes an activity source, an activity target, and an association between the activity source and the activity target;" 89

1. *Capalik's* plurality of activities performed at the first computer system 90
2. *Capalik's* activity sources and targets 91
3. *Capalik's* associations 94

D. Claim [1(c)]: "storing in the memory a data structure that identifies the activity sources, the activity targets, and the associations for the plurality of activities; and" 96

E. Claim [1(d)]: "transmitting to one or more computer systems other than the first computer system information identifying one or more of the activity sources, the activity targets, and the associations for preventing future attacks, to the one or more computer systems, associated with the one or more of the activity sources, the activity targets, and the associations" 103

1. *'872 Patent's* Description of Fingerprint Generation 104
2. *Capalik's* Signature/Fingerprint Generation 104

F. Claim 2: "The method of claim 1, wherein the monitoring further comprises monitoring all activity on the first computer system." 112

G. Claim 3: "The method of claim 1, further comprising: creating, from the stored information, a fingerprint indicative of the activity on the first computer system." 113

H. Claim 4: "The method of claim 1, wherein the activities comprise two or more of: a file initiating an execution of an instruction, an instruction reading a file, an instruction writing to a file, and receiving data through the network" 114

I. Claim 5: "The method of claim 1, wherein a respective activity target comprises one of: a file, a user, an instruction, a thread, data stream, and network socket connections" 116

J. Claim 6: "The method of claim 1, wherein a respective activity source comprises one of: a file, a user, an instruction, a thread, data stream, and network socket connections " 116

K. Claim 12: "The method of claim 1, further comprising: identifying activity sources that are affected by unauthorized activities" 117

L. Claim 15: "The method of claim 12, wherein the identifying activity sources that are affected by unauthorized activities includes identifying data stream incoming from outside the first computer system." 118

M. Claim [20(Pre)]: "A first computer system, comprising: one or more processors; and memory storing one or more programs, the one or more programs including instructions for:".....119

N. Claim [20(a)]: "monitoring activity on the first computer system;".....120

O. Claim [20(b)]: "identifying a plurality of activities being performed at the first computer system, where each of the activities includes an activity source, an activity target, and an association between the activity source and the activity target;".....120

P. Claim [20(c)]: "storing in the memory a data structure that identifies the activity sources, the activity targets, and the associations for the plurality of activities; and".....120

Q. Claim [20(d)]: "transmitting to one or more computer systems other than the first computer system information identifying one or more of the activity sources, the activity

targets, and the associations for preventing future attacks, to the one or more computer systems, associated with the one or more of the activity sources, the activity targets, and the associations."....121

R. Claim [21(Pre)]: "A computer readable storage medium, including one or more programs for execution by one or more processors of a first computer system, the one or more programs including instructions for:"....121

S. Claim [21(a)]: "monitoring activity on the first computer system;"....121

T. Claim [21(b)]: "identifying a plurality of activities being performed at the first computer system, where each of the activities includes an activity source, an activity target, and an association between the activity source and the activity target;"....122

U. Claim [21(c)]: "storing in memory of the first computer system a data structure that identifies the activity sources, the activity targets, and the associations for the plurality of activities; and....122

V. Claim [21(d)]: "transmitting to one or more computer systems other than the first computer system information identifying one or more of the activity sources, activity targets, and associations for preventing future attacks, to the one or more computer systems, associated with the one or more of the activity sources, the activity targets, and the associations."....122

XI. Ground 2: Capalik in view of Goel renders Claims 1-6, 12, 14-15, 20-21 Obvious....122

A. *Capalik-Goel* teaches identifying and storing "activity sources, activity targets, and associations" as recited in independent Claims 1, 19-20....123

1. *Goel* Identifies Activity Source, Activity Target, and Association....124
2. Motivation to Combine *Capalik* and *Goel*....127

B. *Capalik-Goel* Renders Claims 2-6, 12, 15, 20, and 21 Obvious....132

C. *Capalik-Goel* Renders Claim 14 Obvious.....132

1. Claim 14: "The method of claim 13, wherein: instructions executed by the one or more processors have respective privilege levels; respective activity sources have respective privilege levels; and the identifying activity sources that are affected by unauthorized activities includes identifying activity sources that request to execute underprivileged instructions." 132

2. *Goel's* Teachings.....132

3. Motivation to Combine *Capalik* and *Goel* 133

XII. Ground 3: Capalik ALONE OR in view of Goel And in further view of King renders Claims 7-11, 14, 16, 18-19 Obvious.....134

A. Claim 7: "The method of claim 1, further comprising: graphically displaying at least a subset of the stored activities." 134

1. *'872 Patent's* graphically displaying a subset of the stored activities.....134

2. *King* teaches graphically displaying a subset of stored activities.....135

3. Motivation to Combine *Capalik-King/Capalik-Goel-King*....140

B. Claim 8: "The method of claim 7, wherein the displaying includes visually distinguishing at least a subset of the associations." 146

C. Claim 9: "The method of claim 1, wherein an activity target of a first activity of the plurality of activities is an activity source of a second activity of the plurality of activities."....148

1. '872 Patent's Teachings 148

2. *King's* Teachings.....149

3. Motivation to Combine *Capalik/Capalik-Goel* per *King*.....150

D. Claim 10: "The method of claim 1, further comprising: identifying a respective association between a first activity source and a first activity target as unauthorized, wherein the first activity target, when associated as a second activity source with a second activity target, causes unauthorized activities on the second activity target." 151

1. *King's* Teachings 151

2. Motivation to Combine *Capalik/Capalik-Goel* per *King* 153

E. Claim 11: "The method of claim 10, further comprising: tracking a transfer of malicious associations over time." 154

F. Claim 14: "The method of claim 12, wherein: instructions executed by the one or more processors have respective privilege levels, respective activity sources have respective privilege levels; and the identifying activity sources that are affected by unauthorized activities includes identifying activity sources that request to execute underprivileged instructions." 155

1. *King's* teachings 156

2. Motivation to Combine *Capalik/Capalik-Goel* and *King* 157

G. Claim 16: "The method of claim 1, further comprising: determining an activity status for the stored events, the activity status comprising one of: a complete status, representing that a source of unauthorized activities is identified, and each activity target in the stored activities is associated with the source through one or more associations; and an incomplete status, which represents that the source of unauthorized activities is not identified, or at least one activity target in the stored activities is not associated with the source through one or more associations." 158

1. '872 Patent's Description of Complete Status and Incomplete Status 158

2. *King's* Teachings 159

3. Motivation to Combine *Capalik/Capalik-Goel* per *King* 161

H. Claim 18: "The method of claim 1, further comprising: identifying a respective activity as unauthorized based on a determination that the respective activity is associated with an unauthorized activity that is distinct from the respective activity." 161

I. Claim 19: "The method of claim 1, wherein the association includes a type that represents whether or not the activity

source has modified the activity target to perform the unauthorized activities." 162

XIII. Ground 4: Capalik ALONE OR in view of Goel, and in further view of Pike renders Claim 13 Obvious 164

A. Claim 13: "The method of claim 12, wherein the identifying activity sources that are affected by unauthorized activities includes identifying activity sources that request to access a portion of memory that is set as non-executable." 164

1. *Pike's* teachings 164
2. Motivation to Combine *Capalik-Pike/Capalik-Goel-Pike* 166

XIV. Ground 5: Capalik ALONE OR in view of Goel, and in further view of Farley renders Claim 17 Obvious 169

A. Claim 17: "The method of claim 1, further comprising: determining an activity level representing a frequency of unauthorized activities on the first computer system during a predefined time interval." 169

1. *Farley's* teachings 170
2. Motivation to Combine *Capalik-Farley/Capalik-Goel-Farley* 173

XV. CONCLUSION 178

CLAIM LISTING

Claim 1:

Claim 1[Pre]: A computer implemented method of identifying unauthorized activities on a first computer system attached to a computer network, wherein the first computer system comprises one or more processors and memory, the method comprising:

1(a) monitoring activity on the first computer system;

1(b) identifying a plurality of activities being performed at the first computer system, wherein each of the activities includes an activity source, an activity target, and an association between the activity source and the activity target;

1(c) storing in the memory a data structure that identifies the activity sources, the activity targets, and the associations for the plurality of activities; and

1(d) transmitting to one or more computer systems other than the first computer system information identifying one or more of the activity sources, the activity targets, and the associations for preventing future attacks, to the one or more computer systems, associated with the one or more of the activity sources, the activity targets, and the associations.

Claim 2:

The method of claim 1, wherein the monitoring further comprises monitoring all activity on the first computer system.

Claim 3:

The method of claim 1, further comprising: creating, from the stored information, a fingerprint indicative of the activity on the first computer system.

Claim 4:

The method of claim 1, wherein the activities comprise two or more of: a file initiating an execution of an instruction, an instruction reading a file, an instruction writing to a file, and receiving data through the network.

Claim 5:

The method of claim 1, wherein a respective activity target comprises one of: a file, a user, an instruction, a thread, data stream, and network socket connections.

Claim 6:

The method of claim 1, wherein a respective activity source comprises one of: a file, a user, an instruction, a thread, data stream, and network socket connections.

Claim 7:

The method of claim 1, further comprising: graphically displaying at least a subset of the stored activities.

Claim 8:

The method of claim 7, wherein the displaying includes visually distinguishing at least a subset of the associations.

Claim 9:

The method of claim 1, wherein an activity target of a first activity of the plurality of activities is an activity source of a second activity of the plurality of activities.

Claim 10:

The method of claim 1, further comprising: identifying a respective association between a first activity source and a first activity target as unauthorized, wherein the first activity target, when associated as a second activity source with a second activity target, causes unauthorized activities on the second activity target.

Claim 11:

The method of claim 10, further comprising: tracking a transfer of malicious associations over time.

Claim 12:

The method of claim 1, further comprising: identifying activity sources that are affected by unauthorized activities.

Claim 13:

The method of claim 12, wherein the identifying activity sources that are affected by unauthorized activities includes identifying activity sources that request to access a portion of memory that is set as non-executable.

Claim 14:

The method of claim 12, wherein: instructions executed by the one or more processors have respective privilege levels; respective activity sources have respective privilege levels; and the identifying activity sources that are affected by unauthorized activities includes identifying activity sources that request to execute underprivileged instructions.

Claim 15:

The method of claim 12, wherein the identifying activity sources that are affected by unauthorized activities includes identifying data stream incoming from outside the first computer system.

Claim 16:

The method of claim 1, further comprising: determining an activity status for the stored events, the activity status comprising one of: a complete status, representing that a source of unauthorized activities is identified, and each activity target in the stored activities is associated with the source through one or more associations; and an incomplete status, which represents that the source of

unauthorized activities is not identified, or at least one activity target in the stored activities is not associated with the source through one or more associations.

Claim 17:

The method of claim 1, further comprising: determining an activity level representing a frequency of unauthorized activities on the first computer system during a predefined time interval.

Claim 18:

The method of claim 1, further comprising: identifying a respective activity as unauthorized based on a determination that the respective activity is associated with an unauthorized activity that is distinct from the respective activity.

Claim 19:

The method of claim 1, wherein the association includes a type that represents whether or not the activity source has modified the activity target to perform the unauthorized activities.

Claim 20:

Claim 20[Pre]: A first computer system, comprising:

20(a) one or more processors; and

20(b) memory storing one or more programs, the one or more programs including instructions for:

20(c) monitoring activity on the first computer system;

20(d) identifying a plurality of activities being performed at the first computer system, where each of the activities includes an activity source, an activity target, and an association between the activity source and the activity target;

20(e) storing in the memory a data structure that identifies the activity sources, the activity targets, and the associations for the plurality of activities; and

20(f) storing in the memory the activity sources, activity targets, and associations;

20(g) transmitting to one or more computer systems other than the first computer system information identifying one or more of the activity sources, the activity targets, and the associations for preventing future attacks, to the one or more computer systems, associated with the one or more of the activity sources, the activity targets, and the associations

Claim 21:

Claim 21[Pre]: A computer readable storage medium, including one or more programs for execution by one or more processors of a first computer system, the one or more programs including instructions for:

21(a) monitoring activity on the first computer system;

21(b) identifying a plurality of activities being performed at the first computer system, where each of the activities includes an activity source, an activity target, and an association between the activity source and the activity target;

21(c) storing in memory of the first computer system a data structure that identifies the activity sources, the activity targets, and the associations for the plurality of activities; and

21(d) transmitting to one or more computer systems other than the first computer system information identifying one or more of the activity sources, activity targets, and associations for preventing future attacks, to the one or more computer systems, associated with the one or more of the activity sources, the activity targets, and the associations.

I, Markus Jakobsson, declare as follows:

I. INTRODUCTION AND QUALIFICATIONS

1. I am over the age of 21 and am competent to make this declaration. I make this declaration based on facts and matters within my own knowledge and on information provided to me by others, and, if called as a witness, I could and would competently testify to the matters set forth herein.

A. Educational Background and Professional Experience

2. My qualifications are stated more fully in my curriculum vitae, which is attached as Appendix A. Below is a summary of my education, work experience, and other qualifications.

3. I received a Master of Science degree in Computer Engineering from the Lund Institute of Technology in Sweden in 1993, a Master of Science degree in Computer Science from the University of California at San Diego in 1994, and a Ph.D. in Computer Science from the University of California at San Diego in 1997, specializing in Cryptography. During and after my Ph.D. studies, I was also a researcher at the San Diego Supercomputer Center and General Atomics, where I did research on authentication, privacy, and security. Much of my work was on distributed protocols to enable security and privacy, and my Ph.D. thesis described a distributed electronic payment system.

4. Since earning my doctorate degree over twenty-six years ago, I have been an employee, entrepreneur, and consultant in the computer systems, security, and the anti-fraud industry. I have worked extensively with technologies related to computer security, mobile security, phishing, malware detection, quantitative and qualitative fraud analysis, and disruptive security. I have worked as an engineer, researcher or technical advisor at a number of leading companies in the computer industry, including PayPal, Qualcomm, Bell Labs, and LifeLock. I have also taught and performed research at several prestigious universities, including New York University (NYU) and Indiana University (IU). Further, I have been materially involved in designing, developing, testing, and assessing technologies for computer and network security, digital rights management, trust establishment, randomness, cryptography, socio-technical fraud, malware detection, detection of malicious emails, user interfaces, authentication, and fraud detection for over twenty years.

5. From 1997 to 2001, I was a Member of Technical Staff at Bell Labs, where I did research on authentication, privacy, multi-party computation, contract exchange, digital commerce including crypto payments, and fraud detection and prevention. An example publication of mine from this time period includes "Secure distributed computation in cryptographic applications" (U.S. Patent No. 6,950,937, 2001 priority date.) At the end of my tenure at Bell Labs, I had started to research

identity theft years before the banking industry became aware of the existence of phishing attacks.

6. From 2001 to 2004, I was a Principal Research Scientist at RSA Labs, where I worked on predicting future fraud scenarios in commerce and authentication and developed solutions to those problems. Much of my work related to the development of anti-fraud mechanisms, addressing authentication abuse and identity theft. During that time, I predicted the rise of what later became known as phishing. I also laid the groundwork for what I termed "proof of work," and described how this could be applied in the context of distributed computations such as mining of crypto payments; this work later came to influence the development of BitCoin. I was also an Adjunct Associate Professor in the Computer Science department at New York University from 2002 to 2004, where I taught cryptographic protocols, with an emphasis on authentication.

7. From 2004 to 2016, I held a faculty position at Indiana University at Bloomington, first as an Associate Professor of Computer Science, Associate Professor of Informatics, Associate Professor of Cognitive Science, and Associate Director of the Center for Applied Cybersecurity Research (CACR) from 2004 to 2008; and then as an Adjunct Associate Professor from 2008 to 2016. I was the most senior security researcher at Indiana University, where I built a research group focused on online fraud and countermeasures, resulting in over 50 publications and

two books. One of these books, "Crimeware: Understanding New Attacks and Defenses" (Wiley, 2008), described malware-based attacks, online abuse in general, and countermeasures to attacks and abuses of these types.

8. While a professor at Indiana University, I was also employed by Xerox PARC, PayPal, and Qualcomm to provide thought leadership to their security groups. I was a Principal Scientist at Xerox PARC from 2008 to 2010, a Director and Principal Scientist of Consumer Security at PayPal from 2010 to 2013, a Senior Director at Qualcomm from 2013 to 2015, Chief Scientist at Agari from 2016 to 2018, and Chief of Security and Data Analytics at Amber Solutions from 2018 to 2020.

9. I was hired by Qualcomm as they acquired a startup I founded, FatSkunk. FatSkunk had developed algorithms for retroactive detection of malware, with applications to low-power platforms such as mobile devices. Example publications include "Retroactive Detection of Malware With Applications to Mobile Platforms" by Markus Jakobsson and Karl-Anders Johansson, "Practical and Secure Software-Based Attestation" by Markus Jakobsson and Karl-Anders Johansson, and U.S. Patent 8,370,935, titled "Auditing a Device". My work at Qualcomm related to authentication of users and detection of malware, and the integration of the FatSkunk technology in Qualcomm products.

10. Agari is a cybersecurity company that develops and commercializes technology to protect enterprises, their partners and customers from advanced email phishing attacks. At Agari, my research studied and addressed trends in online fraud, especially as related to email, including problems such as Business Email Compromise, Ransomware, and other abuses based on social engineering and identity deception. My work primarily involved identifying trends in fraud and computing before they affected the market, and developing and testing countermeasures, including technological countermeasures, user interaction and education. Example publications include "Detecting computer security risk based on previously observed communications" (U.S. Patent No. 10,715,543 B2, 2016 priority date) and "Using message context to evaluate security of requested data" (U.S. Patent No. 10,805,314 B2, 2017 priority date).

11. In 2020, after leaving Agari, I was the Chief of Security and Data Analytics at Amber Solutions, an IoT startup in the San Francisco Bay Area. Amber Solutions is a cybersecurity company that develops home and office automation technologies. At Amber Solutions, my research addressed privacy, user interfaces and authentication techniques in the context of ubiquitous and wearable computing. My work often related to security in constrained computing environments, as exemplified in "Configuration and management of smart nodes with limited user interfaces" (U.S. Patent No. 10,887,447).

12. I left Amber Solutions to take the role of Chief Scientist at ByteDance in 2019, where I guided the security research both for ByteDance and TikTok until I left to co-found Artema Labs, where I am the Chief Scientist. My work at ByteDance involved identifying new threats, developing new solutions, and guiding research in areas relating to cryptography and Internet security.

13. I am also the Chief Technology Officer at ZapFraud, a cybersecurity company that develops techniques to detect deceptive emails, such as Business Email Compromise ("BEC") emails. At ZapFraud, my research studies and addresses computer abuse, including social engineering, malware and privacy intrusions, and has also involved studying security aspects related to cloud computing. My work primarily involves identifying risks, developing protocols and user experiences, and evaluating the security of proposed approaches.

14. I also work as an independent security researcher and consultant in the fields of computer/mobile security, fraud detection/prevention, digital rights management, malware, phishing, crimeware, mobile malware, and cryptographic protocols. In addition, I am a visiting research fellow of the Anti-Phishing Working Group, an international consortium focused on unifying the global response to cybercrime in the public and private sectors.

15. I have also served on the advisory board of several companies, including Metaforic, a company that developed technology for watermarking

software and provided security software with a built-in framework that resists attacks from malware, bots, hackers and other attempts. Metaforic was acquired by Inside Secure in 2014.

16. I have founded or co-founded several successful computer security companies. In 2005, I co-founded RavenWhite Security, a provider of authentication solutions, and I am currently its Chief Technical Officer. RavenWhite owns intellectual property related to authentication, such as "Performing authentication" (U.S. Patent No. 10,079,823 2007 priority date). In 2007, I co-founded Extricatus, one of the first companies to address consumer security education. In 2009, I founded FatSkunk, a provider of mobile malware detection software; I served as Chief Technical Officer of FatSkunk from 2009 to 2013, when FatSkunk was acquired by Qualcomm, and I became a Qualcomm employee. FatSkunk developed anti-virus technology. In 2013, I founded ZapFraud—the same company described above, where I currently serve as Chief Technical Officer. In 2014, I co-founded RightQuestion, a security consulting company.

17. I have additionally served as a member of the fraud advisory board at LifeLock (an identity theft protection company); a member of the technical advisory board at CellFony (a mobile security company); a member of the technical advisory board at PopGiro (a user reputation company); a member of the technical advisory board at MobiSocial dba Omlet (a social networking company); and a member of

the technical advisory board at Stealth Security (an anti-fraud company). I have provided anti-fraud consulting to KommuneData (a Danish government entity), J.P. Morgan Chase, PayPal, Boku, and Western Union.

18. I have authored seven books and over 100 peer-reviewed publications and have been a named inventor on over 200 patents and patent applications. A large number of these relate to Internet security, authentication, malware, spoofing and related topics.

19. Examples of textbooks I have authored or co-authored on the topic of Internet security include:

- "Crimeware: Understanding New Attacks and Defenses" (Symantec Press, 2008)
- "Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft" (Wiley, 2006);
- "The Death of the Internet" (Wiley, 2012);
- "Understanding Social Engineering Based Scams" (Springer 2016);
- "Towards Trustworthy Elections: New Directions in Electronic Voting" (Springer, 2010).

20. Books translated to Chinese and Korean have separate titles and years of release.

21. I have authored numerous publications regarding Internet security, including Internet security and malware detection. Relevant examples include:

- Tamper-Evident Digital Signatures: Protecting Certification Authorities Against Malware Jong Youl Choi, Philippe Golle, and Markus Jakobsson;
- "Remote Harm-Diagnostics", (PDF) Privacy-preserving history mining for web browsers (researchgate.net);
- "Server-Side Detection of Malware Infection", NSPW Proceedings on New Security Paradigms Workshop (2009).

22. An exemplary relevant patent publication includes U.S. Patent No. 8,578,174. This patent was assigned to Palo Alto Research Center Incorporated arising out of my employment.

23. My opinions are based on my years of education, research, and experience, as well as my study of relevant materials. In forming my opinions, I have also considered the materials identified in this declaration and in the Petition.

24. In sum, I have extensive experience both as a developer and a researcher relating to computer security.

25. I am being compensated for my time at my ordinary hourly rate of $800 My compensation is not dependent on the outcome of these proceedings or the content of my opinions. To the best of my knowledge, I have no financial interest in either party or in the outcome of this proceeding.

II. METHODOLOGY; MATERIALS CONSIDERED

26. I have relied upon my education, knowledge and experience with the technology of communication networks, as well as the other materials as discussed in this declaration in forming my opinions.

27. For this work, I have been asked to review U.S. Patent No. 9,954,872 ("*'872 Patent*") (Ex. 1001) including the specification and claims, and the '872 Patent's prosecution history ("*'872 File History*") (Ex. 1002). In developing my opinions relating to the '872 Patent, I have considered the materials cited or discussed herein, including those listed below.

Exhibit 1001	U.S. Patent No. 9,954,872 (the "*'872 Patent*")
Exhibit 1002	File History for U.S. Patent 9,954,872 (the "*'872 File History*")
Exhibit 1003	*Intentionally left blank*
Exhibit 1004	U.S. Application Publication No. 2008/0016570 to Capalik ("*Capalik*")
Exhibit 1005	Ashvin Goel et al. "*Forensix: A Robust, High-Performance Reconstruction System*," Proceedings of the 25th International Conference on Distributed Computing Systems Workshops, 2005 ("*Goel*")
Exhibit 1006	Samuel T. King, "*Analyzing Intrusions Using Operating System Level Information Flow,"* Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in The University of Michigan (2006) ("*King*")
Exhibit 1007	U.S. Patent No. 8,819,822 to Pike et al. ("*Pike*")
Exhibit 1008	U.S. Patent No. 7,089,428 to Farley ("*Farley*")
Exhibit 1009	*Intentionally left blank*
Exhibit 1010	*Intentionally left blank*
Exhibit 1011	WO 2004/097584 to Knight ("*Knight*")
Exhibit 1012	U.S. Application Publication No. 2006/0242701 to Black et al. ("*Black*")
Exhibit 1013	U.S. Application Publication No. 2007/0208822 to Wang et al. ("*Wang*")

Exhibit 1014	U.S. Application Publication No. 2008/0098476 to Syversen ("*Syversen*")
Exhibit 1015	U.S. Patent No., 7,483,972 to Bhattacharya et al. ("*Bhattacharya*")
Exhibit 1016	U.S. Application Publication No. 2005/0193429 to Demopoulos et al. ("*Demopoulos*")
Exhibit 1017	WO 2009/032379 to Stolfo et al. ("*Stolfo*")
Exhibit 1018	U.S. Application Publication No. 2004/0103314 to Liston ("*Liston*")
Exhibit 1019	U.S. Patent No., 8,407,788 to Kestner et al. ("*Kestner*")
Exhibit 1020	U.S. Application Publication No. 2009/0037672 to Colbert et al. ("*Colbert*")
Exhibit 1021	WO 2009/097610 to Moffie et al. ("*Moffie*")
Exhibit 1022	U.S. Patent No., 8,510,827 to Leake et al. ("*Leake*")
Exhibit 1023	U.S. Patent No., 9,098,698 to Ghosh et al. ("*Ghosh*")
Exhibit 1024	U.S. Application Publication No. 2007/0220518 to Verbowski et al. ("*Verbowski*")
Exhibit 1025	*Intentionally left blank*

28. I have considered these materials from the viewpoint of a POSITA as of the priority date of the '872 Patent. For the purposes of this declaration, I have been asked to assume that the priority date of the '872 Patent is June 24, 2010. I note that my opinions provided in this declaration are made from the perspective of a POSITA as of this priority date of the '872 Patent, unless expressly stated otherwise. To the extent that I use any verb tense in this declaration that is present tense (e.g., "a POSITA would understand" instead of "a POSITA would have understood"), such verb tense should be understood to be my opinion as of the '872 Patent's priority date (again, unless expressly stated otherwise). I merely use the present verb tense for ease of reading.

III. OVERVIEW AND LEGAL STANDARDS

29. In formulating my opinions, I have been instructed to apply certain legal standards. I am not a lawyer. I do not expect to offer any testimony regarding what the law is. Instead, the following sections summarize the law as I have been instructed to apply it in formulating and rendering my opinions found later in this declaration. I understand that, in an *inter partes* review ("IPR") proceeding, patent claims may be deemed unpatentable if it is shown that they were anticipated or rendered obvious in view of the prior art. I understand that prior art in an IPR review is limited to patents or printed publications that predate the priority date of the patent at issue. I understand that questions of claim clarity (definiteness) and enablement cannot be considered as a ground for considering the patentability of a claim in these proceedings.

A. Person of Ordinary Skill in the Art

30. I understand that the '872 Patent, the record of proceedings at the Patent Office (which I understand is called the "File History" or "Prosecution History"), and the teachings of the prior art are evaluated from the perspective of a person of ordinary skill in the art ("POSITA") at the time the claimed invention was made. I understand that the factors considered in determining the ordinary level of skill in the art include: (i) the levels of education and experience of persons working in the field; (ii) the types of problems encountered in the field; and (iii) the sophistication

of the technology. I may also consider, if available, the education level of the inventor, prior art solutions to the problems encountered in the art, and the rapidity with which innovations are made in the relevant art.

31. I understand that a person of ordinary skill in the art is not a specific real individual, but rather a hypothetical individual having the qualities reflected by the factors above. This hypothetical person has knowledge of all prior art in the relevant field as if it were arranged on a workshop wall and takes from each reference what it would teach to a person having the skills of a POSITA.

B. Obviousness

32. I understand that a claim may be invalid under 35 U.S.C. § 103(a) if the subject matter described by the claim as a whole would have been "obvious" to a hypothetical POSITA in view of a single prior art reference or in view of a combination of references at the time the claimed invention was made. Therefore, I understand that obviousness is determined from the perspective of a hypothetical POSITA. I further understand that a hypothetical POSITA is assumed to know and to have all relevant prior art in the field of endeavor covered by the patent in suit and all analogous prior art. I understand that obviousness in an IPR review proceeding is evaluated using a preponderance of the evidence standard, which means that the claims must be more likely obvious than nonobvious.

33. I also understand that an analysis of whether a claimed invention would have been obvious should be considered in light of the scope and content of the prior art, the differences (if any) between the prior art and the claimed invention, and the level of ordinary skill in the pertinent art involved. I understand as well that a prior art reference should be viewed as a whole. I understand that in considering whether an invention for a claimed combination would have been obvious, I may assess whether there are apparent reasons to combine known elements in the prior art in the manner claimed in view of interrelated teachings of multiple prior art references, the effects of demands known to the design community or present in the marketplace, and/or the background knowledge possessed by a POSITA. I also understand that other principles may be relied on in evaluating whether a claimed invention would have been obvious, and that these principles include the following:

- A combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results;
- When a device or technology is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or in a different one, so that if a POSITA can implement a predictable variation, the variation is likely obvious;
- If a technique has been used to improve one device, and a POSITA would have recognized that it would improve similar devices in the

same way, using the technique is obvious unless its actual application is beyond his or her skill;

- An explicit or implicit teaching, suggestion, or motivation to combine two prior art references to form the claimed combination may demonstrate obviousness, but proof of obviousness does not depend on or require showing a teaching, suggestion, or motivation to combine;
- Market demand, rather than scientific literature, can drive design trends and may show obviousness;
- In determining whether the subject matter of a patent claim would have been obvious, neither the particular motivation nor the avowed purpose of the named inventor controls;
- One of the ways in which a patent's subject can be proved obvious is by noting that there existed at the time of invention a known problem for which there was an obvious solution encompassed by the patent's claims;
- Any need or problem known in the field of endeavor at the time of invention and addressed by the patent can provide a reason for combining the elements in the manner claimed;
- "Common sense" teaches that familiar items may have obvious uses beyond their primary purposes, and in many cases a POSITA will be

able to fit the teachings of multiple patents together like pieces of a puzzle;

- A POSITA is also a person of ordinary creativity, and is not an automaton;
- A patent claim can be proved obvious by showing that the claimed combination of elements was "obvious to try," particularly when there is a design need or market pressure to solve a problem and there are a finite number of identified, predictable solutions such that a POSITA would have had good reason to pursue the known options within his or her technical grasp; and
- One should not use hindsight in evaluating whether a claimed invention would have been obvious.

34. I also understand that an obviousness determination includes the consideration of various factors such as: (1) the scope and content of the prior art, (2) the differences between the prior art and the asserted claims, (3) the level of ordinary skill in the pertinent art, and (4) the existence of secondary considerations such as commercial success, long-felt but unresolved needs, failure of others, and so forth.

35. I am informed that it is improper to combine references where the references teach away from their combination. I am informed that a reference may

be said to teach away when a person of ordinary skill in the relevant art, upon reading the reference, would be discouraged from following the path set out in the reference, or would be led in a direction divergent from the path that was taken by the patent applicant. In general, a reference will teach away if it suggests that the line of development flowing from the reference's disclosure is unlikely to be productive of the result sought by the patentee. I am informed that a reference teaches away, for example, if (1) the combination would produce a seemingly inoperative device, or (2) the references leave the impression that the product would not have the property sought by the patentee. I also am informed, however, that a reference does not teach away if it merely expresses a general preference for an alternative invention but does not criticize, discredit, or otherwise discourage investigation into the invention claimed.

36. I am informed that even if a *prima facie* case of obviousness is established, the final determination of obviousness must also consider "secondary considerations" if presented. In most instances, the patentee raises these secondary considerations of non-obviousness. In that context, the patentee argues an invention would not have been obvious in view of these considerations, which include: (a) commercial success of a product due to the merits of the claimed invention; (b) a long-felt, but unsatisfied need for the invention; (c) failure of others to find the solution provided by the claimed invention; (d) deliberate copying of the invention

by others; (e) unexpected results achieved by the invention; (f) praise of the invention by others skilled in the art; (g) lack of independent simultaneous invention within a comparatively short space of time; and (h) teaching away from the invention in the prior art.

37. I am further informed that secondary considerations evidence is only relevant if the offering party establishes a connection, or nexus, between the evidence and the claimed invention. The nexus cannot be based on prior art features. The establishment of a nexus is a question of fact. While I understand that Patent Owner here has not offered any secondary considerations at this time, I will supplement my opinions in the event that Patent Owner raises secondary considerations during the course of this proceeding.

C. Analogous Art

38. I have been informed that for a prior art reference to be proper for use in an obviousness analysis, the reference must be "analogous art" to the claimed invention. I have been informed that a reference is analogous art to the claimed invention if: (1) the reference is from the same field of endeavor as the claimed invention (even if it addresses a different problem); or (2) the reference is reasonably pertinent to the problem faced by the inventor (even if it is not in the same field of endeavor as the claimed invention). In order for a reference to be "reasonably pertinent" to the problem, it must logically have commended itself to an inventor's

attention in considering his problem. In determining whether a reference is reasonably pertinent, one should consider the problem faced by the inventor, as reflected either explicitly or implicitly, in the specification. I believe that all of the references that my opinions in this IPR are based upon are well within the range of references a person having ordinary skill in the art would consult to address the type of problems described in the Challenged Claims.

D. Claim Construction

39. I understand that the claim terms in this IPR will be construed according to their plain and ordinary meaning as understood in light of the claim language, the patent's description, and the prosecution history viewed from the perspective of a POSITA. I further understand that where a patent defines claim language, the definition in the patent controls, even if there are other definitions that might be understood by those working in the art.

40. I understand the Petition proposes the Board adopt the interpretation of "association between the activity source and the activity target" as the ***action***[1] occurring between the activity source and activity target. Accordingly, I have also adopted Petitioner's interpretation of "association between the activity source and

[1] An action can be any event or operation, including calling, executing, modifying, or accessing, such as system calls and library calls. *'872 Patent*, 13:34-38.

the activity target" in my opinions below, and I take no position as to which is correct. For all other claim language, I have applied the plain and ordinary meaning as defined in the preceding paragraph.

IV. LEVEL OF A PERSON OF ORDINARY SKILL

41. Based on my review and analysis of the '872 Patent, the prior art cited herein, and the ordinary skill factors described in this section, a POSITA in the field of the '872 Patent at the time of the earliest possible priority date (June 24, 2010) would have been knowledgeable regarding the field of software security analysis and dynamic malware analysis. In my experience working in this field, most workers of ordinary skill in the art as of the earliest possible priority date of June 24, 2010, would have had a bachelor's degree in computer science, computer engineering, or an equivalent, as well as two years of industry experience and who would have had a working knowledge of software security analysis and dynamic malware analysis. *See, e.g., '872 Patent* (Ex. 1001), 1:44-48, 2:9-11, 2:27-48, 4:37-5:9, 5:18-30, 7:59-67. A person with less relevant practical experience but with additional education can also qualify as a POSITA in the field of the '872 Patent. When I refer to the understanding of a POSITA, I am referring to the understanding of such a person as of June 24, 2010.

42. As of June 24, 2010, I had more than ordinary skill in the art. I am, however, familiar with the skills and knowledge possessed by those I would have

considered to be of ordinary skill in the art as of that date. For example, I worked on applied security such as detection of and countermeasures against phishing and malware. In 2008, I published a textbook, "Crimeware" (Wiley, ISBN-13: 978-0321501950) that explored the trends in malware-based abuse and described useful countermeasures. In 2009, I co-founded FatSkunk, a startup that developed novel malware countermeasures; FatSkunk was acquired by Qualcomm in 2013. I also published extensively on malware detection, e.g., my 2010 publication on detection of zero-day malware infections, "Retroactive Detection of Malware with Applications to Mobile Platforms" *USENIX conference on Hot topics in security* (2010). A zero-day attack is one of which there does not exist any prior knowledge, such as behavioral attack patterns or relevant signatures.

43. My opinions provided in this declaration would not change in view of minor modifications to this level of ordinary skill.

V. OVERVIEW OF THE TECHNOLOGY

A. Intrusion Detection Systems

44. Intrusion Detection/Prevention Systems (IDS/IPS) were well known before the priority date of the '872 Patent. In general, intrusion detection systems were used to protect network sources from malicious attacks. As part of this process, Intrusion Detection Systems are normally connected to a network to, for example, monitor incoming traffic on the network connection. For example, *Knight* teaches

an "intrusion detection capability generally includes three functional components: (1) a stream source that provides chronological event information; (2) an analysis mechanism that determines potential or actual intrusions; and (3) a response mechanism that takes action on the output of the analysis mechanism." *Knight* (Ex. 1011), [0036].

45. In one example of an IDS, *Black* teaches "session level auditing is also performed to understand the network traffic and accordingly identify security vulnerabilities on the network system and check for potential methods that an intruder might use to attack the network system resources." *Black* (Ex. 1012), [0014]. *Black* also teaches learning new attack patterns; that is, if an attack vector does not match previously identified attack patterns, the new attack pattern can be analyzed and learned. *Black*, [0018]. After learning the new attack pattern, *Black* teaches a signature of the attack is generated.

> "If attack vector 110 is not identified, learning module 212 is initiated. In one embodiment of the invention, learning module 212 understands how attack vector 110 is occurring within network system 100. This is performed by emulating all the network system resources and services on network device 102, if the type of attack vector 110 is not identified. Further, learning module 212 determines which network system resource is being attacked and how attack vector 110 is collecting information. Depending on the understanding of the attack, learning

> module 212 generates a new signature that corresponds to attack vector 110."

Black, [0018].

46. Performing such analysis allows an administrator to protect the network and devices on the network from future attacks. Specifically, performing such analysis after an attack allows an administrator to recreate an attack, analyze what actions occurred, and then create a signature to prevent the attack from occurring again in the future. In this way, it was known in the prior art to develop defenses against even new attacks.

> "One of the possible uses of embodiments of the invention from the audit policy standpoint is the ability to re-create an attack after the fact. Through intelligent logging of what signatures are being provided based on the user inputs, embodiments of the invention can quickly re-create an attack session. A possible output of the audit would be a network traffic log which could be read by a network sniffer application for analysis. From that log one could see who did what to compromise the system being attacked—the first step in providing a patch. Without embodiments of the invention, the system administrator is only left with 'clues' (e.g., syslog events, corrupted server from the attack, etc) based on what limited information is available after the fact. Embodiments of the present invention allow the capturing of what has actually happened, including identifying the particular attacker."

Black, [0021], [0024]-[0025] (teaching generating signatures from attack vectors, including a "signature that represents the response of network system resources to attack vector 110 … The signatures are then used to emulate the required network system resources and services, if the same attack vector is transmitted again."), [0031] (teaching using a learning mode to learn a previously unknown attack pattern).

47. One particular type of attack was known as a "zero-day" exploit. *Wang* teaches that a "zero-day exploit is considered to be an exploit that exists before the patch for the exploited vulnerability is released. The vulnerability related to the zero-day exploit can be known or unknown generally (e.g., to the public) at the time of detecting the zero-day exploit." *Wang* (Ex. 1013), [0053]. Some intrusion detection systems were designed to protect against such zero-day exploits. For example, *Syversen* and *Wang* both teach defending against zero-day attacks, with *Syversen* focusing on zero-day worm-based attacks. *Syversen* (Ex. 1014), Abstract, [0002], [0034]; *Wang*, [0085], [0090].

48. It was also known that intrusion detection systems were able to log numerous pieces of information. For example, *Bhattacharya* teaches an event log storing information in an "event message" including source address, source port, destination address, destination port, IP protocol, timestamp, event ID, and event type. *Bhattacharya* (Ex. 1015), 3:51-4:60. *Bhattacharya's* Fig. 2 below depicts a

series of such event messages that are stored in a network security monitor's event log:

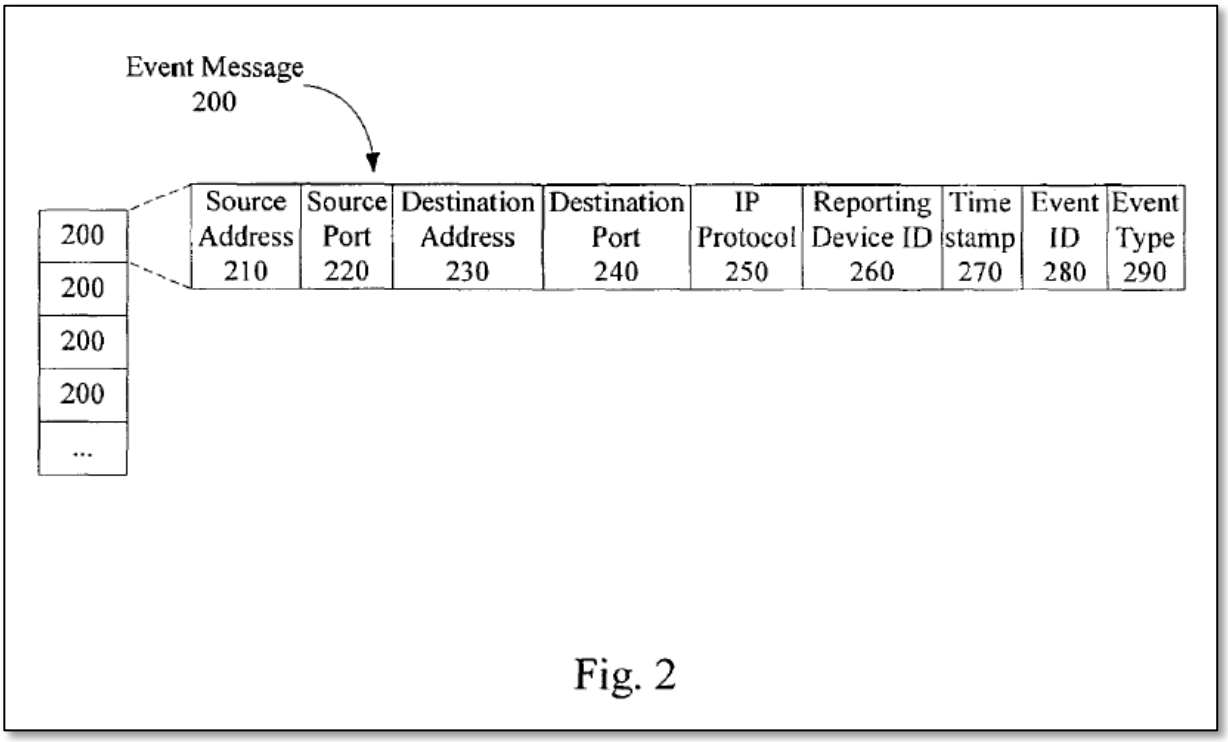


Fig. 2

Bhattacharya, Fig. 2. *Bhattacharya* teaches correlating event messages to monitor various types of attacks. *Bhattacharya*, 4:17-60. Event correlation rules are utilized to detect various security attacks that follow particular patterns. *Bhattacharya*, 5:7-57.

B. Honeypots/Decoy Systems

49. One particular example of an Intrusion Detection System is referred to as a "honeypot" or "decoy system." Honeypots and other decoy systems were well known in the prior art before the '872 Patent. In general, a honeypot is a "method to trap intruders and to track attackers[.]" *Demopoulos* (Ex. 1016), [0022]. In another

example, *Knight* teaches that a "Honeypot is an intrusion detection mechanism that attempts to lure attackers by presenting a more visible and apparently more vulnerable resource than the network itself. Honeypots are useful for detecting attacks, since they provide a single point for security professionals to monitor for evidence of anomalous activity. They are also useful in retaining significant data pertaining to an attack." *Knight*, [0045].

50. More particularly, a honeypot is "used to attract computer attacks to a virtual operational system that is a virtual instantiation of a typical deployed operating system." *Syversen*, Abstract, [0034] ("The honey pot system is specifically configured as a virtual network that is an instantiation of the real network. It is thus created to look and act like the real network. Traffic coming into the system or out of the system exercises processes within the honey pot virtual network, called a honey net, so that non-normal operation is quickly spotted."). *Syversen* teaches using the honeypot to gather information to protect the real network. *Syversen*, [0036], [0038], [0044] (teaching numerous examples of data that are monitored for malicious behavior, including stack changes, register states, malformed packets, port numbers, IP addresses, user account changes such as permissions and new creations, disk activity, and memory usage), [0046]-[0047]. In another example, *Black* teaches that honeypots can track numerous pieces of information, including IP address, MAC address, TCP window size, TCP flags, port, and type of application. *Black*,

[0037]. This information is used to track activities or changes that occur during an attack.

51. *Black* teaches honeypots "emulate network system resources and services, thereby protecting the network system. Honeypots can track and capture the activity that directly interacts with them. Honeypots are part of the individualized hardware that runs the network device." *Black*, [0005], [0017] ("A honeypot is an information system resource whose value lies in unauthorized or illicit use of the resource. A honeypot is a part of the network system, which is isolated and protected."). A honeypot can be integrated with a network-connected device along with other modules for performing attack analysis and creating signatures. *Black*, [0022], [0025] ("an attack vector is detected by a honeypot, and the type of the attack vector is identified. In one embodiment of the invention, a honeypot is triggered to intercept an attack vector by an outside network device, router or administrator.").

52. It was also known that all traffic to a honeypot is unauthorized. *Black* teaches a "honeypot is an information system resource. **Any interaction with honeypots indicates unauthorized or malicious activity.**"[2] *Black*, [0005], [0017] ("**Any interaction with honeypots indicates unauthorized or malicious activity.**").

---

[2] All emphases added unless otherwise noted.

53. Additionally, decoy systems can act as "bait" for an attacker. *Stolfo* teaches "[t]raps using decoy information (sometimes referred to herein as 'bait information,' 'bait traffic,' or 'decoy media') can be set up to attract and/or confuse attackers (both inside and outside attackers) and/or malware." *Stolfo* (Ex. 1017), [0022]. Further, utilizing "unused IP addresses within a network" can also serve to trap or monitor attacker activity. *Liston* (Ex. 1018), [0014]-[0015].

54. In particular, honeypot systems may be used for observation, where administrators may "monitor intruders' activities without repairing the changes or alerting the intruders, in order to better protect against future attacks." *Kestner* (Ex. 1019), 7:27-34. Additionally, activities on the honeypot "would be logged and reported for analysis and to assist in taking corrective action" to protect against intrusions. *Kestner*, 8:30-42.

C. Virtual Machines, Hypervisors/Virtual Machine Monitors

55. It was known to utilize virtual machines and hypervisors/virtual machine monitors for intrusion detection systems before the priority date of the '872 Patent. For example, *Colbert* teaches that "[a]s is well known in the field of computer science, a virtual machine (VM) is an abstraction–a 'virtualization'–of an actual physical computer system" and that a "virtual machine (VM) or 'guest' 200 is installed on a 'host platform,' or simply 'host,' which includes system hardware 100, that is, a hardware platform, and one or more layers or co-resident components

comprising system-level software, such as an operating system (OS) or similar kernel, or a virtual machine monitor, or a hypervisor … or a similar software layer responsible for coordinating and mediating access to hardware resources."). *Colbert* (Ex. 1020), [0059]. In particular, *Colbert* teaches "hypervisor" often refers to a VMM alone or a VMM and a kernel together. *Colbert*, [0064]. *Colbert* teaches "tracking correspondences among data, for example, in a virtual computing environment." *Colbert*, [0002]. Specifically, *Colbert* teaches "virtualization software (for example, a VMM/hypervisor) is in a functional position to monitor data movement among memory locations and disks, the network, I/O devices, etc." *Colbert*, [0020].

56. In another example, *Moffie* teaches "providing intrusion detection by monitoring events in a virtualized environment." *Moffie* (Ex. 1021), [0001]. In particular, such a setup can be utilized to detect zero day or other new exploits or attacks. *Moffie*, [0014] ("The approaches described herein are proactive by design and remain adaptive, identifying and defending against new exploits and attacks as they are deployed."). *Moffie* teaches utilizing a VMM-based IDS (VIDS) that "utilizes the virtual machine monitor (VMM) layer in a virtualized system to extract VMM-level semantics or information during runtime." *Moffie*, [0015]. *Moffie* also teaches that its techniques include extracting "VMM-level events" and developing features that "accurately identify security intrusions in compute-server

applications." *Moffie*, [0018]. *Moffie* teaches that "[a]s known to one of ordinary skill in the art, the applications and the components of the virtual machine 114 are instructions and data stored in memory, just as any other software. The concept, design and operation of virtual machines are well known in the field of computer architecture." *Moffie*, [0029].

57. In particular, it was known that virtual machines provide various advantages for systems. In one example, virtual machines are advantageous because they provide "the ability to run multiple virtual machines on an underlying hardware platform." *Leake* (Ex. 1022), 7:1-3. As another advantage, virtual machines can isolate unsafe/malicious software or activity so that it does not affect the underlying system hardware. *Leake*, 7:7-13 ("Depending on how a particular virtualization system is implemented, it can also provide greater security since the individual virtual machines can isolate potentially unstable or unsafe software so that it cannot adversely affect the hardware state or system files required for running the physical (as opposed to virtual) hardware.").

58. *Moffie* teaches another advantage of using a VM and VMM, which is that "each VM can be isolated from all others, and from all software other than the VMM, which itself will be transparent to the VM; indeed … the user of a VM will be completely unaware that it is not running directly on the hardware, but on a virtualized set of hardware resources." *Moffie*, [0033]. *Moffie* teaches that its system

advantageously protects data, applications, and an operating system from malicious code attacks and insider threats and that obtaining information for its analysis does not introduce significant runtime overhead into the execution environment. *Moffie*, [0035]-[0037]. Further, *Moffie* teaches its VM and VMM system can advantageously capture "all execution, i.e., events, may be captured (versus only application execution events or a subset of the operating system execution events)" which "helps to assure that an intruder cannot identify a potential hole in the present system because all execution may be analyzed and/or captured." *Moffie*, [0062].

59. In another example, *Ghosh* teaches using a virtual machine to detect malicious software. For example, *Ghosh* teaches a guest operating system virtual machine (OS VM) monitors the virtual machine watching for a "trigger event[.]" *Ghosh* (Ex. 1023), 12:54-67, Abstract. If there is a trigger event, *Ghosh* teaches taking steps in response that include sending an activity report to a central appliance with a repository of activities for infected devices. *Ghosh*, 13:1-14.

60. In a further example, *Verbowski* teaches an approach to intercepting and analyzing "threads" including with a virtual machine. *Verbowski* (Ex. 1024), [0027]-[0030], [0037], [0044], [0056]. *Verbowski* discloses monitoring activity, provides some examples of different types of activity, and sends the data to another machine for further analysis. *Verbowski*, [0028]-[0031]. Further, *Verbowski* teaches

generating a "signature" for some monitored activity. *Verbowski*, [0031], [0145]-[0150], [0193]-[0196], [0199]-[0212].

D. Fingerprints/Signatures

61. It was also known that intrusion detection/prevention systems were utilized to generate fingerprints (also referred to as signatures) to store attack patterns for future reference to prevent said attacks from occurring again. One example is *Black*, which teaches a "method and system for preventing, auditing and trending unauthorized traffic in a network system." *Black*, [0002]. *Black* also teaches a "learning phase is used to generate a new signature and record the sequence of events initiated by the current unknown activity." *Black*, [0014], [0017] ("Resource emulator 208 also includes a learning engine 212, hereinafter referred to as learning module 212. Signature creator 202 creates signatures for network system resources and services on network device 102. Each signature represents the response of a network system resource and service available on the network device to an attack vector."), [0017] ("The signatures are stored in system of record 204. In an embodiment of the invention, system of record 204 may be a database.").

62. In another example of generating signatures based on attack data involving a honeypot system, *Syversen* teaches monitoring honeypot network activity to determine "the characteristics of the attack and to put out timely threat data and new rules, settings, tables, signatures and patterns to follow on protection

purposes." *Syversen*, [0062]. Also, *Verbowski* teaches generating a signature indicating rogue activity on a system. *Verbowski*, [0031], [0145]-[0150], [0193]-[0196], [0199]-[0212].

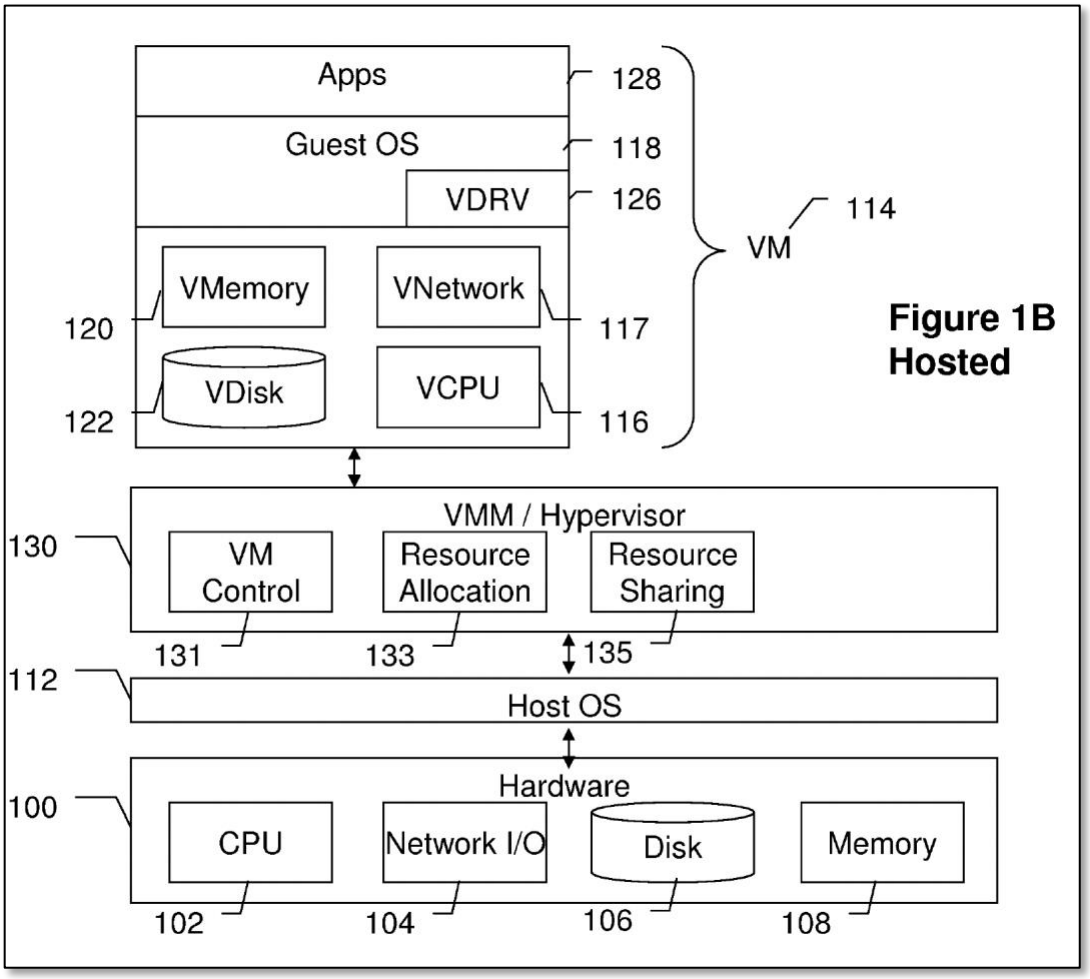
63. *Moffie* teaches deriving "features" from events and provides "the information in an input format for processing by the back-end." *Moffie*, [0090]. *Moffie's* features "capture characteristics present in the event stream" and "do quantify patterns that are effective in identifying normal and/or abnormal execution. Using features rather than events can help identify relevant execution patterns and behaviors." *Moffie*, [0090]-[0091], [0109]-[0120] (teaching both off-line modeling and on-line modeling).

E. Intrusion Detection System Hardware

64. It was known before the priority date of the '872 Patent that IDS/IPS included various hardware, such as one or more computers, one or more processor(s), memory (non-volatile and/or volatile), and hardware for enabling network connections. In one example, *Black* teaches a device includes a "general-purpose computer," a router, and a network system such as a local area network (LAN) or wide area network (WAN) and virtual area network (VLAN). *Black*, [0015].

65. In another example, *Moffie* teaches a VMM-based IDS that includes a central processor (CPU) 102 that "may be a single processor, or multiple cooperating

processors in a known multiprocessor arrangement." *Moffie*, [0025]. Further, *Moffie* teaches that "[a]s in other known systems, the hardware includes, or is connected to, memory 107, disk 106 and network I/O 104." *Id.; see also* Figs. 1A-1B.



Moffie, Fig. 1B.

66. It was also known that intrusion detection systems included hardware for graphically displaying event logs and analysis results. For example, *Bhattacharya* teaches network security monitor system 300 may include user interface 304, which includes display 306 and keyboard 308. *Bhattacharya*, 4:27-

41, Fig. 3. In particular, *Bhattacharya's* Fig. 3 below depicts hardware of network security monitor 300 including CPU 302, memory 314, and user interface 304 (which includes display 306 and keyboard 308):

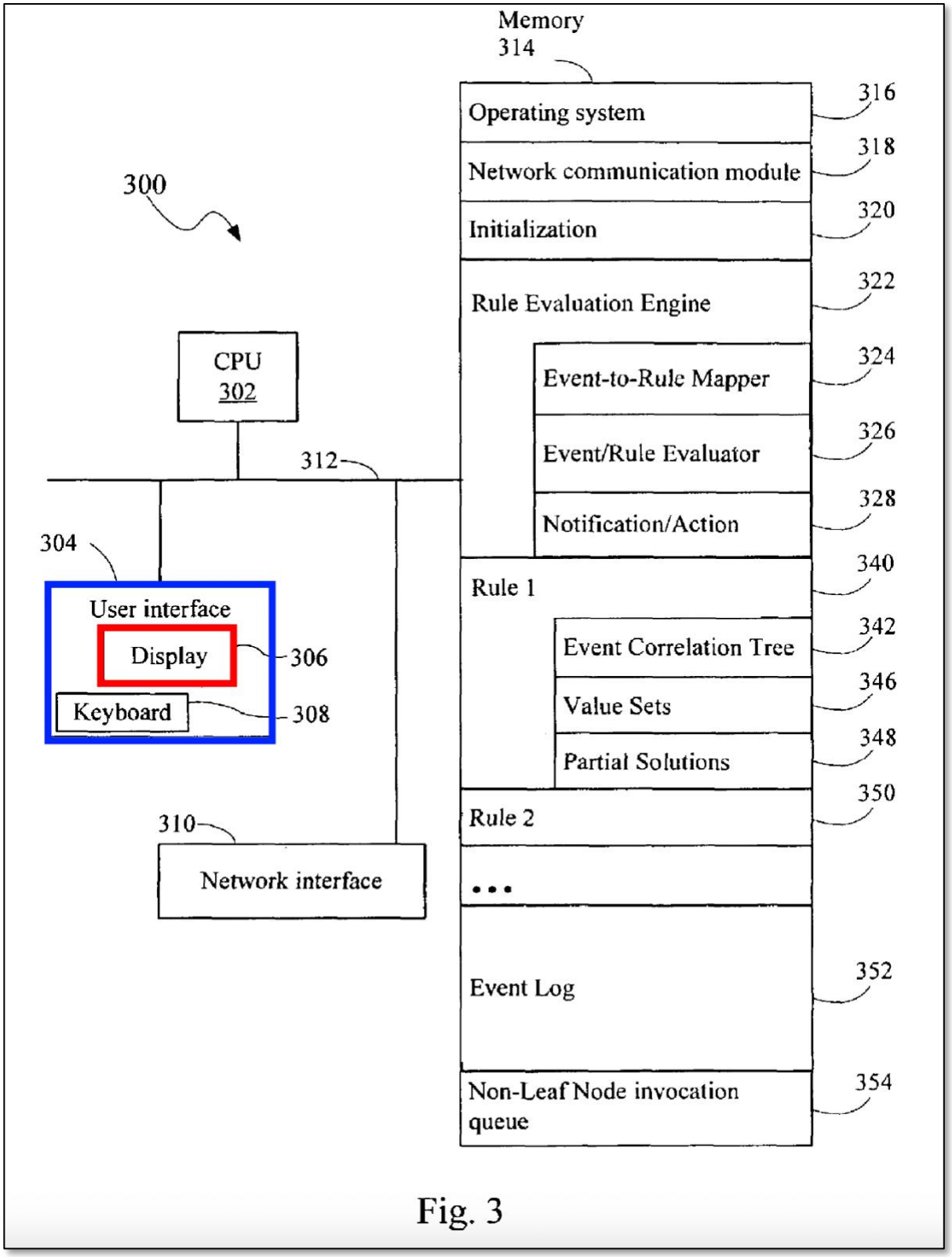


Fig. 3

Bhattacharya, Fig. 3.

VI. OVERVIEW OF THE ’872 PATENT

67. I have reviewed the ’872 Patent (Ex. 1001). Descriptions of the ’872 Patent that may be pertinent to my analysis are provided below. This discussion does not represent my entire understanding of the ’872 Patent.

68. The ’872 Patent describes “methods for analyzing malicious activities on a computer system in order to better protect the computer system from future malicious activity.” *’872 Patent*, 1:44-48. More specifically, the ’872 Patent describes a method for using a virtual machine (VM) as a decoy computer system to monitor unauthorized activities and to generate a fingerprint indicative of those unauthorized activities. *Id.*, 2:27-48, Fig. 1. The fingerprint can then be used to protect the remainder of the computer network against similar unauthorized activities. *Id.*, 2:42-48. By generating fingerprints from monitored activities in real-time, the ’872 Patent purports to protect against so-called “zero-day attack[s]”—security vulnerabilities previously unknown to software developers or system operators. *Id.*, 2:9-11.

69. An exemplary distributed computer system embodying these methods is depicted below in Figure 1:

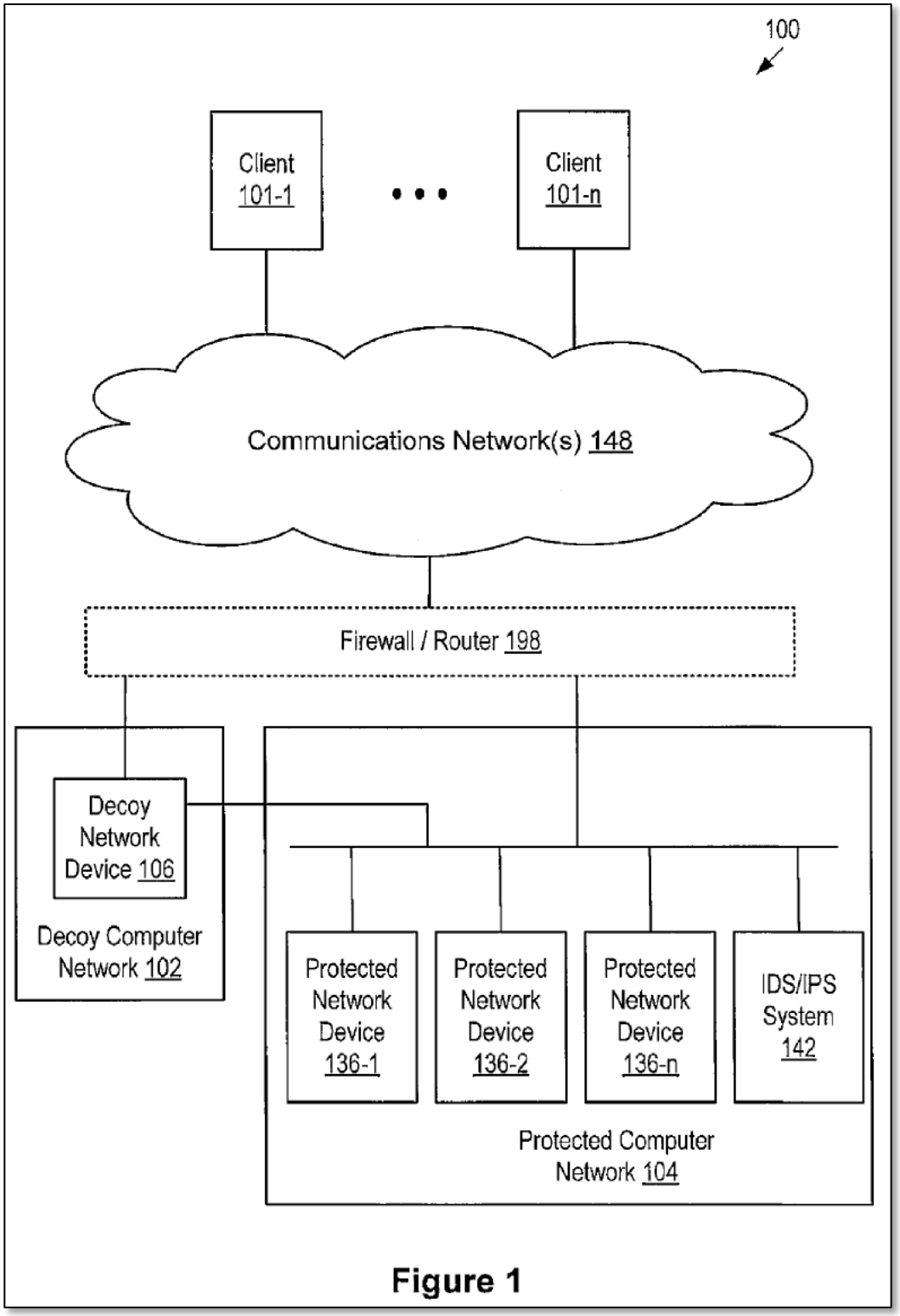

100
Client 101-1
Client 101-n
Communications Network(s) 148
Firewall / Router 198
Decoy Network Device 106
Decoy Computer Network 102
Protected Network Device 136-1
Protected Network Device 136-2
Protected Network Device 136-n
IDS/IPS System 142
Protected Computer Network 104

Figure 1

Id., Fig. 1. As depicted, the disclosed system includes a "decoy computer network 102, a communications network 148, and protected computer network 104." *Id*., 4:5-8. The decoy computer network 102 comprises a "decoy network device 106" used "to collect fingerprint data of unauthorized activities." *Id*., 4:57-60. This decoy network device is "intentionally kept vulnerable to unauthorized or malicious activities" to "allow attackers to attack the decoy network device 106, so that the pattern of the attack can be monitored and analyzed to generate a fingerprint" and prevent similar future attacks. *Id*., 4:60-5:9.

70. Figure 2 below depicts an exemplary decoy network device 106:

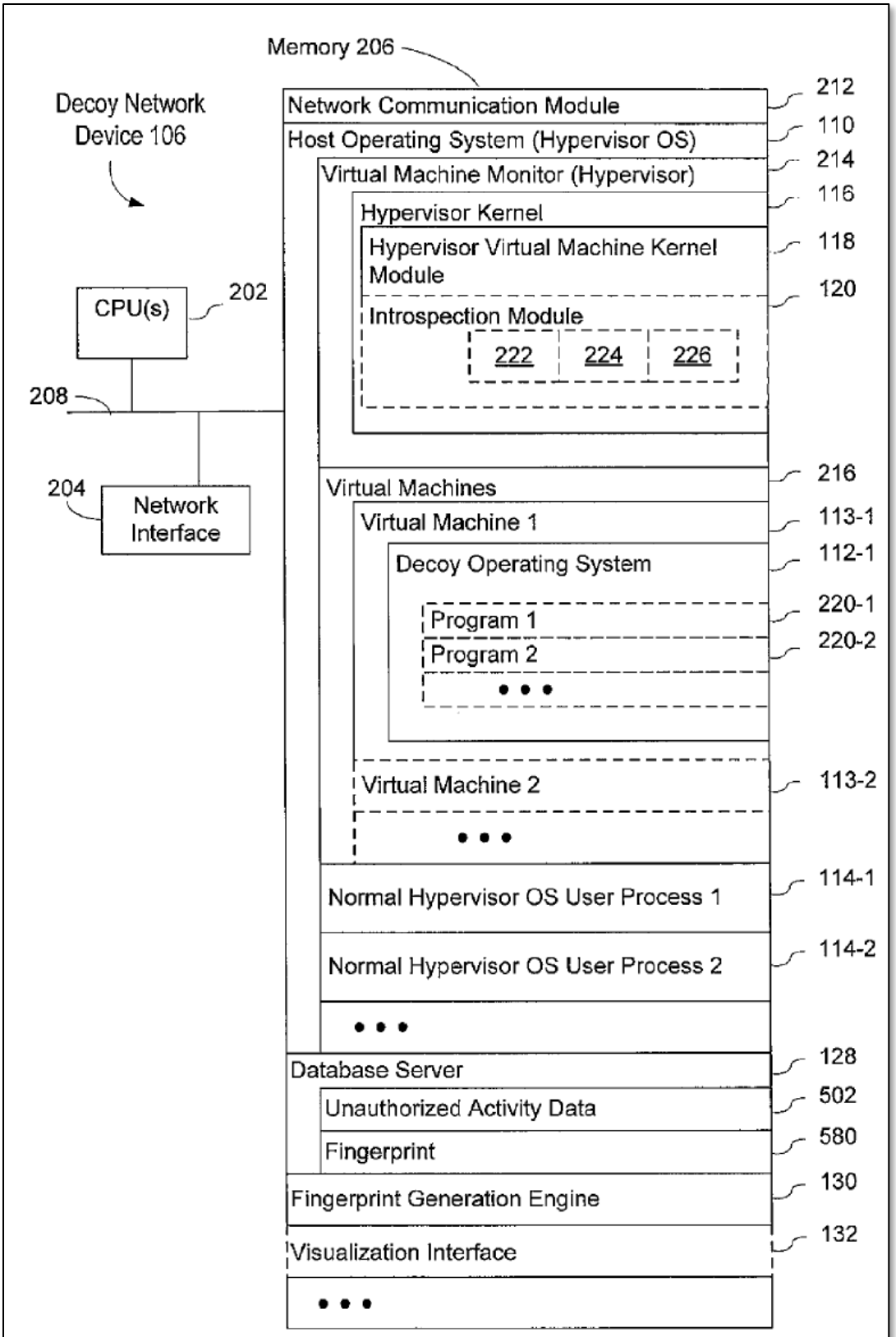

Memory 206
Decoy Network Device 106
Network Communication Module
212
Host Operating System (Hypervisor OS)
110
Virtual Machine Monitor (Hypervisor)
214
Hypervisor Kernel
116
Hypervisor Virtual Machine Kernel Module
118
Introspection Module
120
222
224
226
CPU(s)
202
208
204
Network Interface
Virtual Machines
216
Virtual Machine 1
113-1
Decoy Operating System
112-1
Program 1
220-1
Program 2
220-2
Virtual Machine 2
113-2
Normal Hypervisor OS User Process 1
114-1
Normal Hypervisor OS User Process 2
114-2
Database Server
128
Unauthorized Activity Data
502
Fingerprint
580
Fingerprint Generation Engine
130
Visualization Interface
132

Figure 2

Id., Fig. 2. Decoy network device 106 includes one or more processing units (CPUs) 202, memory 206, a host operating system (e.g., the host OS 110), a database server 128, a fingerprint generation engine 130, and a visualization interface 132. *Id*., 5:31-6:17. The host operating system includes a virtual machine monitor 214, virtual machines 216, and normal hypervisor OS user processes (114-1 and 114-2). *Id*., 6:24-34. The virtual machine monitor 214 includes an introspection module 120 interposed between the virtual machine monitor 214 and decoy operating system(s) 112. *Id*., 6:35-52. The introspection module captures "raw attack information" (i.e., unauthorized activity data) and communicates this information for storage into a "database server 128 as forensic data 236 for later analysis." *Id*., 10:30-33. The fingerprint generating engine 130 then uses this raw forensic information 236 to generate a signature of the attack. *Id*., 10:33-36. The IDS/IPS system 142 uses the fingerprint to identify and prevent subsequent attacks. *Id*., 10:59-61.

71. Figs. 3A-3D and 4A-4E illustrate exemplary patterns of attack monitored on a VM. *Id*., 11:15-22 The network diagram of the pattern of attack depicted in Figs. 3A-3D starts with the detection of an unauthorized activity—Socket 306 receiving data Packet Stream 304. *Id*., 11:26-36. Once the first detected activity is deemed to be "affected by unauthorized activities," related unauthorized activities are identified, providing a detailed picture of the compromise. *Id*., 11:45-47, Figs. 3B-3D.

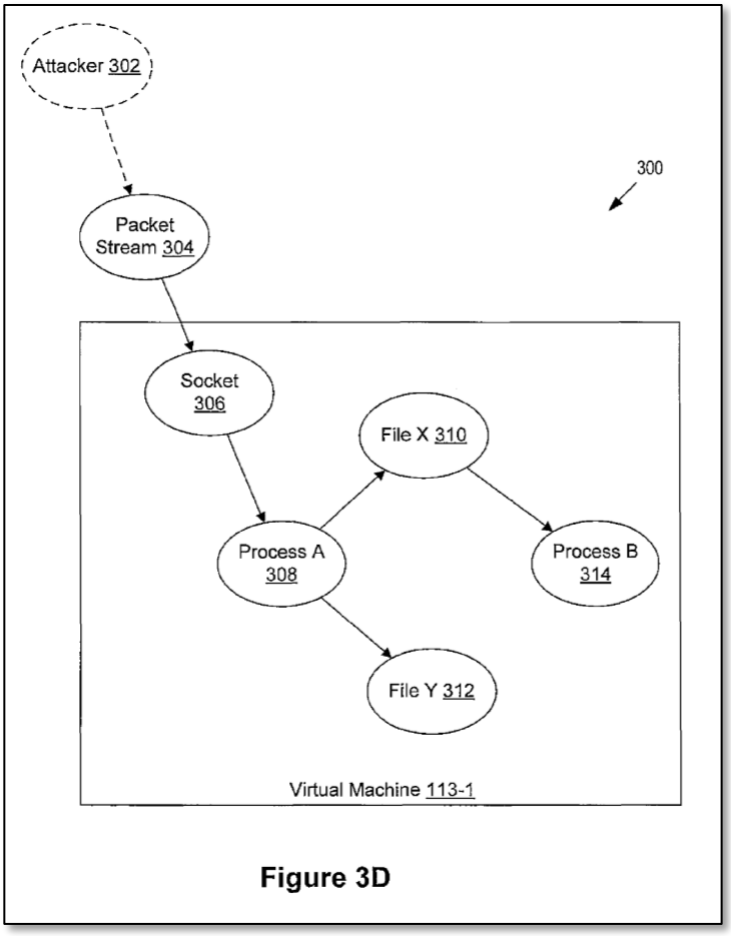


Figure 3D

Id., Fig. 3D, 11:23-12:26. The attack pattern identified in Figs. 4A-4D is virtually identical to the one depicted in Figs. 3A-3D. However, analysis of unauthorized activities in Figs. 4A-4D starts with the detection of unauthorized activities in File X 310, and File Y 312 (e.g., modification of the files by an unauthorized process), and related unauthorized activities are identified backwards in time, providing a picture of how an attacker broke into the system. *Id*., 11:66-12:26, Figs. 4A-4D.

72. The ’872 Patent explains that by analyzing the activities depicted in Figures 3A-3D and 4A-4D, “the fingerprint of the unauthorized activities can be identified[.]” *Id.*, 11:56-57. “By using this information, the security of the protected

network devices 136 can be improved (e.g., the socket 306 can be closed; the process A 308 can be modified to not respond to data similar to the packet stream 304; etc.)." *Id.*, 11:61-65.

VII. THE PROSECUTION HISTORY OF THE '872 PATENT

73. I have reviewed the prosecution history for the '872 Patent. *'872 Patent File History* (Ex. 1002). Pertinent aspects of the prosecution history of the '872 Patent are described below.

A. Prosecution History of the '872 Patent

74. The '872 Patent was filed August 11, 2015, claiming priority to Provisional Application No. 61/358,367 filed June 24, 2010. For the purposes of this declaration, I have been asked to assume that the priority date of the '872 Patent is June 24, 2010, but I have not analyzed whether Patent Owner could establish priority based on the provisional application.

75. A first office action issued on August 30, 2016, rejecting all originally presented claims 1-20. *'872 Patent File History* (Ex. 1002), 127-39. In response, Applicant amended the claims to require storing "<u>a data structure that identifies</u> the activity sources, <u>the</u> activity targets, and <u>the</u> associations <u>for the plurality of activities</u>; and transmitting one or more of the activity sources, <u>the</u> activity targets, and <u>the</u> associations to prevent future attacks <u>associated with the one or more of the activity sources, the activity targets, and the associations</u>." *Id.*, 112-15 (amended

language underlined). Applicant removed claim elements directed to a virtual machine. *Id.*

76. A second office action issued on June 6, 2017, rejecting all amended claims 1-20. *Id.*, 70-84. In response, Applicant submitted a Request for Continued Examination (RCE) on November 6, 2017. *Id.*, 49-59. Along with the RCE, Applicant amended the claims to require "transmitting to one or more computer systems other than the first computer system information identifying one or more of the activity sources, the activity targets, and the associations for preventing future attacks, to the one or more computer systems, associated with the one or more of the activity sources, the activity targets, and the associations." *Id.*, 55-58 (amended language underlined). A Notice of Allowance issued December 15, 2017. *Id.*, 10-16.

VIII. SUMMARY OF UNPATENTABILITY

Proposed Grounds of Unpatentability	**Exhibits**
Ground 1: Claims 1-6, 12, 15, and 20-21 of the '872 Patent are obvious under §103(a) over US Application Publication No. 2008/0016570 to Capalik ("*Capalik*")	Ex. 1004
Ground 2: Claims 1-6, 12, 14-15, and 20-21 of the '872 Patent are obvious under §103(a) over *Capalik* in view of "*Forensix: A Robust, High-Performance Reconstruction System"* by Goel et al. ("*Goel*").	Ex. 1004 Ex. 1005
Ground 3: Claims 7-11, 14, 16, and 18-19 of the '872 Patent are obvious under §103(a) over *Capalik* alone or in view of *Goel* and	Ex. 1004

Proposed Grounds of Unpatentability	**Exhibits**
further in view of "*Analyzing Intrusions Using Operating System Level Information Flow"* by King ("*King*").	Ex. 1005 Ex. 1006
Ground 4: Claim 13 of the '872 Patent is obvious under §103(a) over *Capalik* alone or in view of *Goel* and in further view of U.S. Patent No., 8,819,822 to Pike et al. ("*Pike*").	Ex. 1004 Ex. 1005 Ex. 1007
Ground 5: Claim 17 of the '872 Patent is obvious under §103(a) over *Capalik* alone or in view of *Goel* and in further view of US Patent No., 7,089,428 to Farley et al. ("*Farley*").	Ex. 1004 Ex. 1005 Ex. 1008

77. Above, I have reproduced the Proposed Grounds of Unpatentability from the Petition for ease of reference.

78. As I opine below, *Capalik* teaches a virtual machine and virtual machine monitor for monitoring activity on the virtual machine. *Capalik* teaches that the virtual machine monitor identifies a plurality of activities being performed at the virtual machine, wherein each of the activities includes an activity source, an activity target, and an association between the activity source and activity target. *Capalik's* association includes an action between the source and target. *Capalik* teaches storing in memory the activity sources, activity targets, and associations. *Capalik* teaches creating, from the stored activities, a fingerprint indicative of the activity on the

virtual machine. *Capalik* teaches transmitting the fingerprint to prevent future attacks that comprise the same or similar activities as indicated by the fingerprint. *See* Ground 1.

79. Ground 2 accounts for the possibility that Patent Owner disputes *Capalik's* teachings regarding activity sources, targets, and associations. *Goel* describes a similar system to *Capalik*, elaborating further on specific data fields stored incident to monitoring attacks. *See* Ground 2.

80. *King* teaches graphically displaying subsets of associations and transfer of associations over time. *King also* teaches privilege levels of instructions and activity statuses. *See* Ground 3. *Pike* teaches identifying activity sources that request to access a portion of non-executable memory. *See* Ground 4. Farley teaches determining a frequency of unauthorized activities during a predefined time interval. *See* Ground 5.

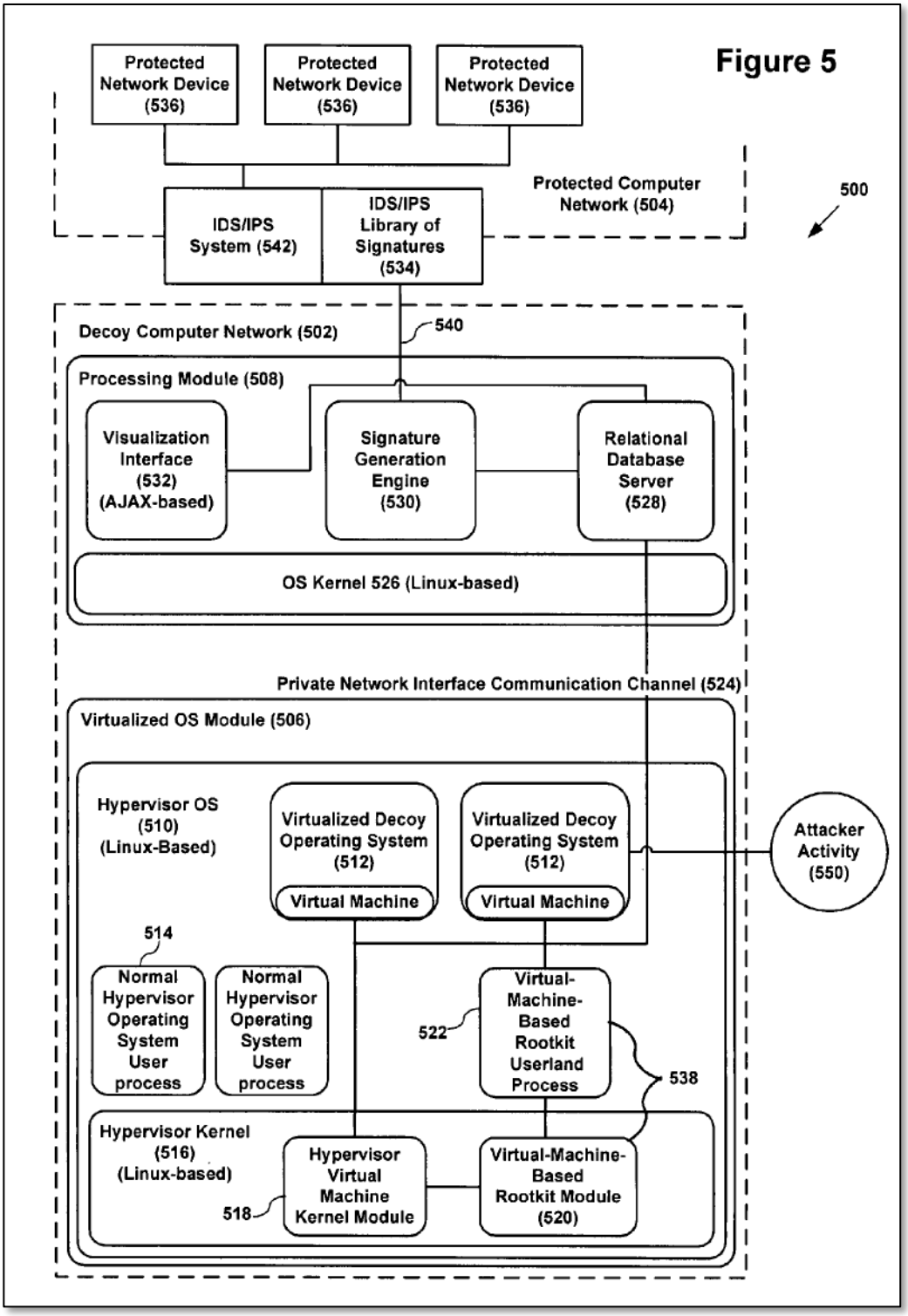
IX. OVERVIEW OF THE PRIOR ART

A. Overview of *Capalik*

81. *Capalik* discloses a method and system for analyzing unauthorized intrusions on a computer system in order to protect a computer network from future unauthorized intrusions. *See Capalik* (Ex. 1004), Abstract, [0002]. More specifically, *Capalik's* method includes using a virtual machine as a decoy computer system to monitor unauthorized activities on the virtual machine, generate a

signature indicative of the attack and protect computer networks from new and undefined attacks (i.e. zero-day attacks). *Id*., Abstract, [0007]-[0011].

82. Figure 5 below depicts an exemplary system for analyzing and preventing unauthorized intrusion into a computer network, embodying *Capalik's* method:



Id., Fig. 5, [0037]. The system shown in Figure 5 includes a "decoy computer network 502 and a protected computer network 504, each comprising one or more

separate computing devices. *Id*., [0038]. Decoy computer network 502 includes virtualized operating system module 506 for monitoring decoy network 502, and processing module 508 for obtaining, analyzing, and responding to exploits."[3] *Id*.

83. *Capalik* explains that one skilled in the art would have appreciated that each of these computing devices may include one or more processors, memory, and any other hardware necessary for hosting and running these modules. *Id.*, [0039]. The virtualized operating system module 506 includes a virtual machine monitor operating system 510, multiple virtual machines 512, and normal hypervisor OS processes 514. The virtual machine monitor operating system 510 includes introspection module 538. *Id.*, [0048]. The processing module 508 includes a signature-generation engine 530, a database server 528, and a visualization interface 532. *Id*., [0051], [0056].

84. The decoy network is intentionally kept vulnerable to lure an attacker. *Id*., [0044]. When an attacker begins interacting with a virtualized decoy operating system, the introspection module 538 monitors and captures attack-identifying information, which is then sent to the processing module 508. *Id.*, [0049]-[0050]. The attack information is stored on the relational database server 528 as forensic

---

[3] *Capalik* explains that these modules may be hosted on the same computing device or on separate computing devices. *Capalik*, [0039].

data for later analysis. *Id.*, [0052]. The processing module 508 generates a report of the attack, containing attack-identifying information, to decide as to whether to generate an attack signature.[4] *Id.*, [0055]. The attack-identifying information is analyzed for known attack patterns as well as non-standard patterns, and the attack may be categorized and analyzed to determine, for example, how an attack gained control of the decoy operating systems(s). *Id.*, [0057]. Graphical representations of key information may also be included in the report. *Id.*, [0056]. The signature-generation engine 530 then uses this forensic data to generate a signature of the attack. *Id.* The IDS/IPS system 542 uses the fingerprint to identify and prevent subsequent attacks. *Id.*

85. As I discussed above, I have been informed that for a prior art reference to be proper for use in an obviousness analysis, the reference must be "analogous art" to the claimed invention. I have been informed that a prior art reference is analogous to the claimed invention if, for example, the reference is from the same field of endeavor as the claimed invention.

---

[4] If the attack signature already exists, or if the data transfer is of a nature that indicates probing rather than attack, then the attack signature may not be generated. *Capalik*, [0059].

86. *Capalik*, like the ’872 Patent, discloses systems and methods for providing computer security by analyzing event data from potential threat events. *Compare Capalik*, Abstract, [0002], [0015] *with ’872 Patent*, 1:44-48, 2:32-3:5. Thus, *Capalik* is in the same field of endeavor as the ’872 Patent.

87. *Capalik* is also reasonably pertinent to certain problems addressed by the ’872 Patent such as providing a forensic analysis of the impact of an attack on a target device. *Compare Capalik*, Abstract, [0002], [0003], [0007], [0015] *with ’872 Patent*, 2:20-31, 10:30-55. Thus, *Capalik* is analogous art to the ’872 Patent.

B. Overview of *Goel*

88. *Goel* discloses a system (Forensix) that analyzes intrusions into a computer system. *Goel* (Ex. 1005), 1. The goal of Forensix is to build a system that supports reliable reconstruction of all computer system activity for forensic purposes. *Id*. To that end, Forensix (1) monitors and captures all system activities after an intrusion, including the cascade of related activities in the system; (2) stores the information in a secure database; and (3) uses database technology to support post-facto analysis based on specific queries about the attack. *Id*. Figure 1 below shows the architecture implementation of *Goel’s* Forensix system:

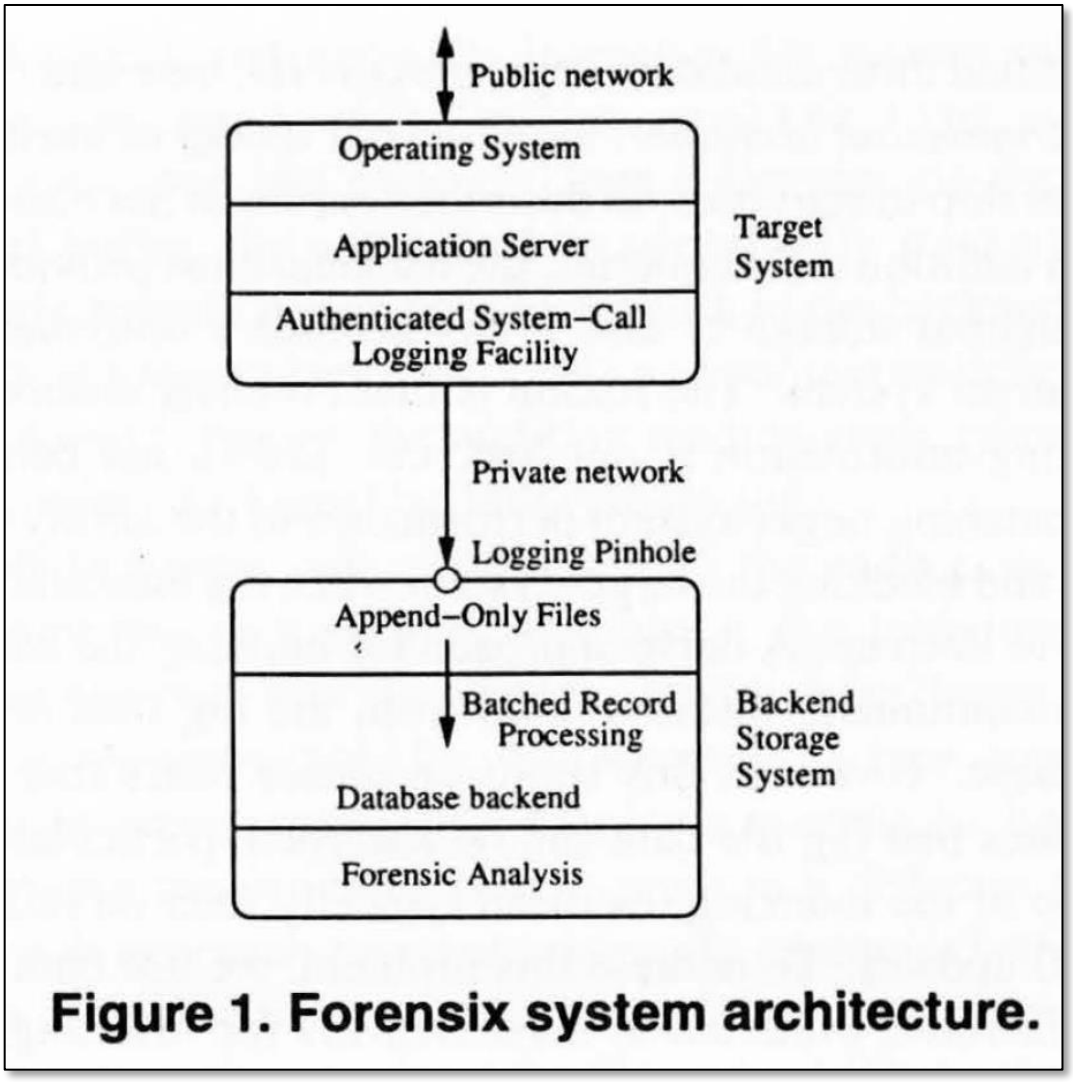


Figure 1. Forensix system architecture.

Goel, Fig. 1. As depicted in Fig. 1, *Goel* teaches Forensix includes a target system that monitors and logs system-calls trace data of an intrusion and a backend storage system that receives the trace data from the network and stores it in a form that allows issuing forensic queries. *Id*., 2, 4. The target system's kernel is instrumented with a logging facility which captures and logs within the kernel all activity across the system-call interface. *Id*., 3. The logging facility then streams system-call traces over a private network interface to a highly secure backend storage system. *Id*., 2.

The backend storage system stores the trace data in a relational database in a form that provides complete DBMS query processing capabilities to run a varied number of queries and scripts that allow replay of system activities for forensic purposes. *Goel* discloses that the Forensix system can respond to a varied number of queries depending on the forensic analysis scenario. *Id*., 1-2. For example, the Forensix system supports queries that allow system administrators to discover where the hacker connected from, how the hacker entered and what the hacker did after he entered or what may have been deleted by the hacker. *Id*., 1.

89. *Goel*, like the ’872 Patent, discloses systems and methods for providing computer security by analyzing event data from potential threat events. *Compare Goel*, 1, 3-4, Fig. 2 *with ’872 Patent*, 1:44-48, 2:32-3:5. Thus, *Goel* is in the same field of endeavor as the ’872 Patent

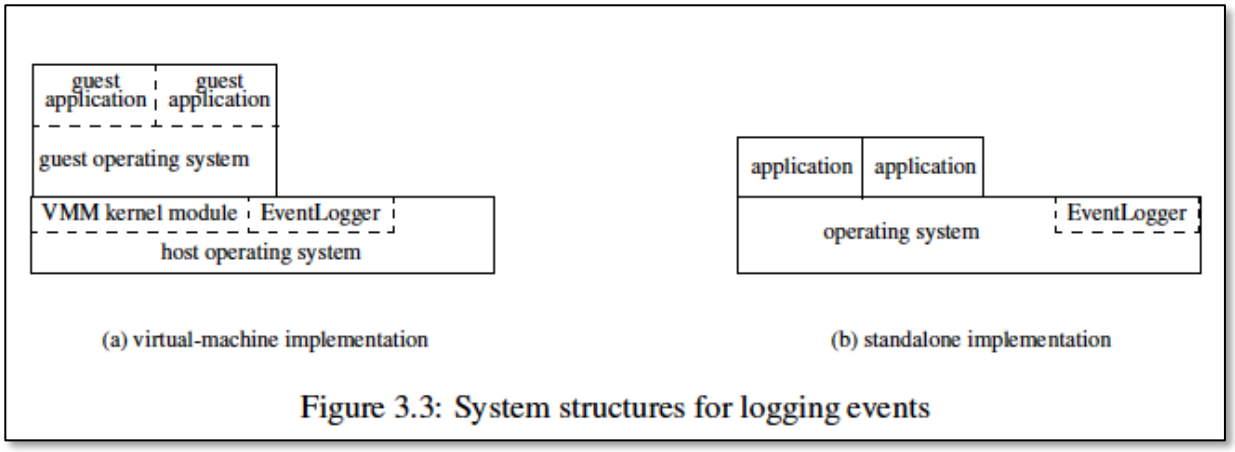
90. *Goel* is also reasonably pertinent to certain problems addressed by the ’872 Patent such as providing a forensic analysis of the impact of an attack on a target device. *Compare Goel*, Introduction, 1, *with ’872 Patent*, 2:20-31, 10:30-55. Thus, *Goel* is analogous art to the ’872 Patent.

C. Overview of *King*

91. *King* teaches systems and methods for analyzing unauthorized intrusions in a computer system to protect the system from future malicious attacks. *King* (Ex. 1006), 2-3, 64-65. Specifically, *King* discloses analyzing unauthorized

activities using "operating-system-level (OS-level) information-flow graphs to highlight the activities of an attacker" being performed at a target system. *King*, 3.

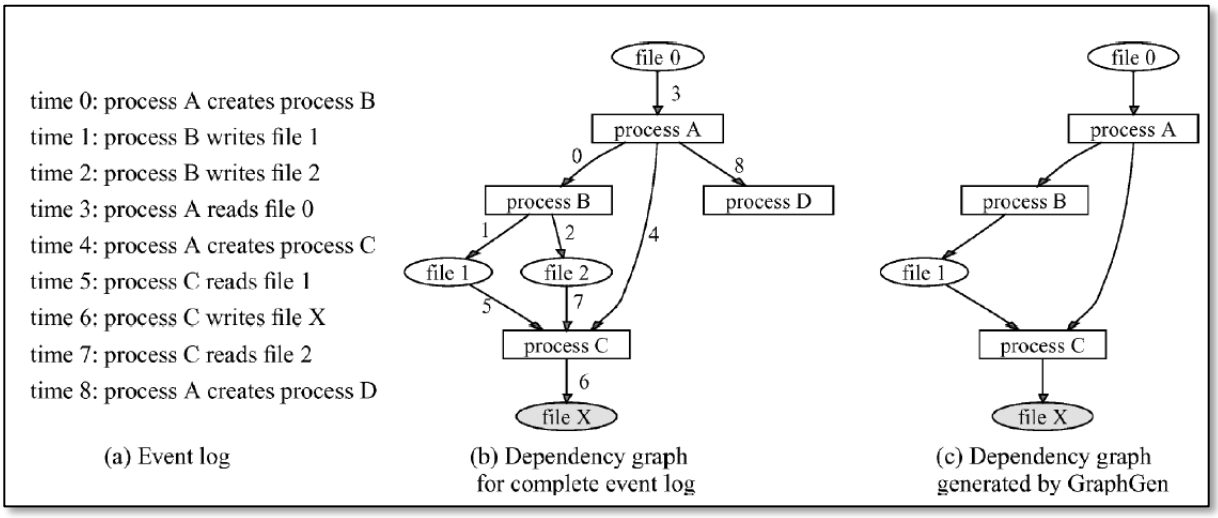
92. *King's* methods for analyzing unauthorized activities can be implemented using a VM or as a standalone implementation, without a VM. *King*, 22-25.



King, Fig. 3.3. *King* uses an "EventLogger" to log information-flow events in real time and "GraphGen" to process EventLogger logs to generate the information-flow graphs of an intrusion. *King*, 12. In the VM implementation depicted in Figure 3.3a, EventLogger examines the state of the VM and obtains information about events and objects inside the target system. The target OS (e.g., Linux 2.4.18), and applications run inside the VM as guest OS and guest applications. *Id*. *King* uses a VM honeypot to test and evaluate its methods by allowing intruders to attack the systems. *King*, 28.

93. *King* discloses three systems and methods for analyzing intrusions using information-flow graphs: (1) "BackTracker[,]" (2) "ForwardTracker," and (3) "Bi-directional Distributed BackTracker (BDB)[.]" *King*, 4, 28-42. BackTracker provides information-flow graphs that provide an analysis of how an intruder broke into a system, and ForwardTracker provides information-flow graphs that provide an analysis of the attacker's actions after the compromise. *Id.* BackTracker's and ForwardTracker's analyses are limited to events that occur on the same host as the detection point. *King*, 42. BDB tracks intrusions as they infect multiple hosts when an attack propagates in a computer network. *Id*. BDB can track intrusions across multiple hosts to find other compromised hosts upstream (using backward causality) and downstream (using forward causality). *Id*. In each of these systems, information-flow graphs start from a "detection point[,]" which refers to the state on the local computer system that alerts the administrator to the intrusion (e.g., a deleted, modified, or additional file, or a process that is behaving in an unusual/suspicious manner). *King*, 4, 17. From the detection point, *King's* systems build a dependency graph of all objects and events that causally affect the state of the detection point (backward information flow) or are causally affected by the state of the detection point (forward information flow). *King*, 17. The objects that *King's* systems track include processes, files, filenames, and registry entries. *King*, 12-13.

94. An exemplary implementation of *King's* Backtracker system is shown in Figure 3.2 below:



King, Fig. 3.2. *King's* BackTracker information-flow graph starts with detection of a suspicious object (i.e., file X having the wrong contents at time 10) and follows the attack back in time, using causal events, to highlight the sequence of events and objects that led to the suspicious state. *King*, 17.

95. GraphGen reads the event logs, starting from the last event and reads toward the beginning of the Event log (i.e., process A creates process B at time 0). *King*, 17. Figure 3.2(b) depicts the event log as a complete dependency graph, including all the events GraphGen considers relevant based on time. *King*, 17-18. GraphGen generates a filtered graph (Fig. 3.2(c)) by pruning away events and object that do not affect file X by time 10. *King*, 18-19. *King* teaches several methods to prioritize or filter information-flow graphs to highlight parts that are most likely to be helpful in understanding intrusions. *King*, 26-28. Information-flow graphs can be

used as "signatures" of known good or bad behavior in computer systems to detect and prevent future intrusions. *King*, 64-65.

96. *King*, like the '872 Patent, discloses systems and methods for providing computer security by analyzing event data from potential threat events. *Compare King*, 2-3 *with '872 Patent*, 1:44-48, 2:32-3:5. Thus, *King* is in the same field of endeavor as the '872 Patent.

97. *King* is also reasonably pertinent to certain problems addressed by the '872 Patent such as providing a forensic analysis of the impact of an attack on a target device. *Compare King*, 2-5, *with '872 Patent*, 2:20-31, 10:30-55. Thus, *King* is analogous art to the '872 Patent.

D. Overview of *Pike*

98. *Pike* discloses computational security techniques for identifying and preventing malware attacks. *Pike* (Ex. 1007), 2:7-8. More specifically, *Pike* teaches improving intrusion detection systems to identify attacks that seek to compromise computer systems by introducing into a target system data that can be interpreted to facilitate attacks (e.g., buffer overflow attacks, internet worms). *Pike*, 1:25-31, 2:6-23, 3:12-21. *Pike* discloses scanning "input strings"[5] (e.g., data packets), identifying

[5] *Pike*, 5:13-38 (teaching input strings as information content such as data packets, supplied by computers 121-123 to computational systems 110).

subsequences in the input strings that could be maliciously exploited, and configuring the target computer system to trigger a fault or action when certain execution conditions suggest use of data or code implicated by the identified subsequences. *Pike*, 3:12-19, 5:13-38, 8:65-67. *Pike* explains a variety of techniques may be employed to configure the target computational system to trigger a fault or action, including setting a relevant no-execute bit, unmapping the corresponding memory page, marking a page as non-readable or non-writable, truncating a code segment, truncating a data segment or setting a breakpoint. *Pike*, 8:67-9:5. Figure 1 below depicts an exemplary network for employing *Pike's* techniques:

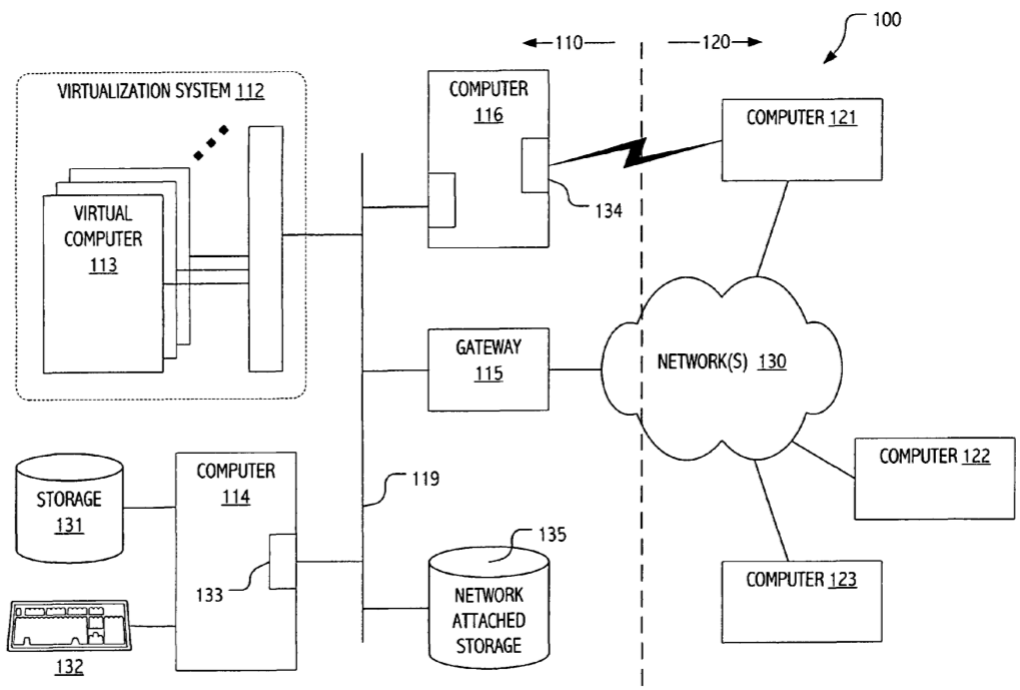


FIG. 1

Pike, Fig. 1. Figure 1 illustrates networked system 100 in which computational systems 110 interface with external and potentially untrusted computers 121-123 (input string sources). *Pike*, 4:59-67. Input string data is presented to one or more computational systems 110 which may include traditional computers 116, 114, virtualization system 112, virtual machines 113, and/or combinations of the foregoing. *Pike*, 5:3-15, 6:19-24.

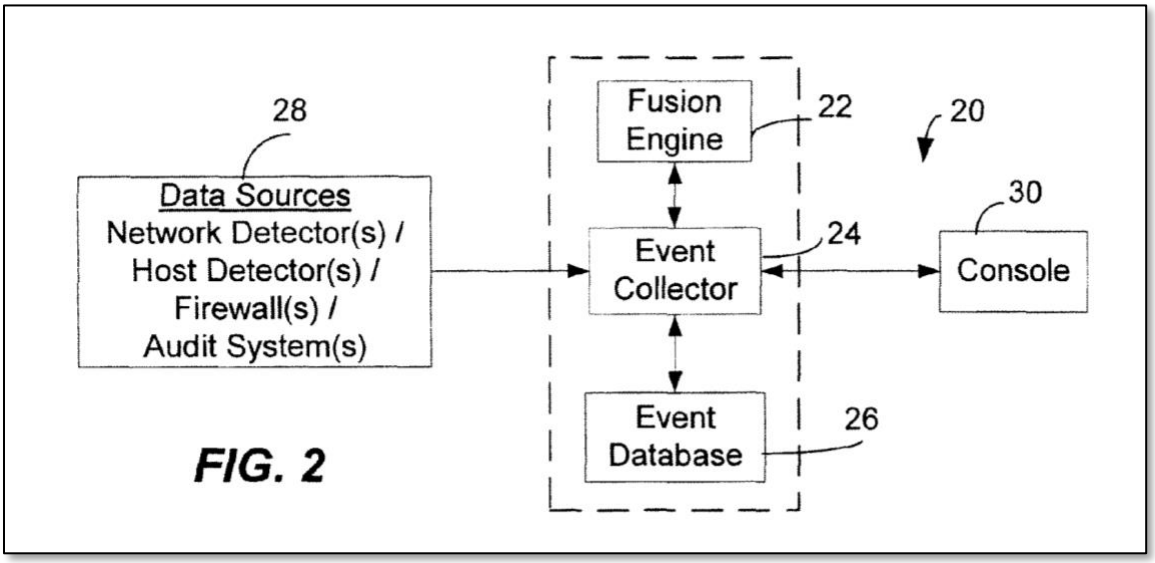
99. *Pike*, like the ’872 Patent, discloses systems and methods for providing computer security by analyzing event data from potential threat events. *Compare Pike*, 2:7-23, 3:10-22 *with ’872 Patent*, 1:44-48, 2:32-3:5. Thus, *Pike* is in the same field of endeavor as the ’872 Patent.

100. *Pike* is also reasonably pertinent to certain problems addressed by the ’872 Patent such as providing a forensic analysis of the impact of an attack on a target device. *Compare Pike*, 2:34-47, 7:30-41 *with ’872 Patent*, 2:20-31, 10:30-55. Thus, *Pike* is analogous art to the ’872 Patent.

E. Overview of *Farley*

101. *Farley* discloses a “computer security management system that can log, investigate, respond to, and track computer security incidents that can occur in a networked computer system. The invention can track suspicious computer activity or actual computer security threats.” *Farley* (Ex. 1008), 3:25-30. More specifically, *Farley* discloses managing security information collected from one or more “data

sources." *Farley*, 3:37-39. Data sources can comprise sensors or detectors that monitor network traffic, individual computers or both, including IDS, firewalls, and other like security devices. *Farley*, 3:47-56. *Farley's* system utilizes a "fusion engine" that fuses or assembles information from multiple data sources and analyzes this information to detect relationships between "raw events" that may indicate malicious behavior, without slowing down processing performed by data sources. *Farley*, 3:39-46. *Farley* refers to "raw events" as suspicious computer activity descriptions obtained from data sources, and actual computer security threats as "mature correlation events." *Farley*, 3:34-37. *Farley's* Figure 2 depicts an exemplary embodiment of system 20 for managing security information collected from one or more data sources:



Farley, Fig. 2.

102. As depicted in Figure 2, security system 20 comprises a fusion engine 22 linked to event collector 24 and event database 26. *Farley*, 9:44-60. Security system 20 comprises data sources 28 and console 30 linked to event collector 24. *Farley*, 9:44-60. Intrusion detection system (IDS) detectors 28 scan raw network traffic or local system events for predefined patterns and generate a "raw event" which is then sent to the event collector and later to fusion engine 22. *Farley*, 10:16-21. The fusion engine assembles raw events received from event collector 24 to provide an organized presentation by correlating raw computer events that are related to each other. *Farley*, 10:20-27.

103. Fusion engine 22 includes a CoBRA (Context-Based Risk Adjustment) processor for the context-based risk adjustment of raw computer events. *Farley*, 14:17-19, Fig. 6. The CoBRA processor may adjust the priority value of raw computer events by comparing a raw computer event against data contained within a context database. *Farley*, 14:19:22. The context database comprises environmental information that may be helpful in evaluating the importance of a raw event, such as historical event frequency, vulnerability of computers within the network, and relative location of a computer. *Farley*, 23:48-60. Once fusion engine 22 determines that two or more events are related to each other (to form a "correlation" event), fusion engine 22 generates messages and forwards these messages to event collector 24. *Farley*, 10:29-32. Event collector 24, in turn, forwards messages fusion engine

22 generates to console 30, which can comprise a program module that runs on a separate personal computer. *Farley*, 10:32-34. Console 30 displays in an organized manner the output of the fusion engine for a user. *Farley*, 3:39-46.

104. *Farley*, like the '872 Patent, discloses systems and methods for providing computer security by analyzing event data from potential threat events. *Compare Farley*, 3:25-46, 4:1-34 *with '872 Patent*, 1:44-48, 2:32-3:5. Thus, *Farley* is in the same field of endeavor as the '872 Patent.

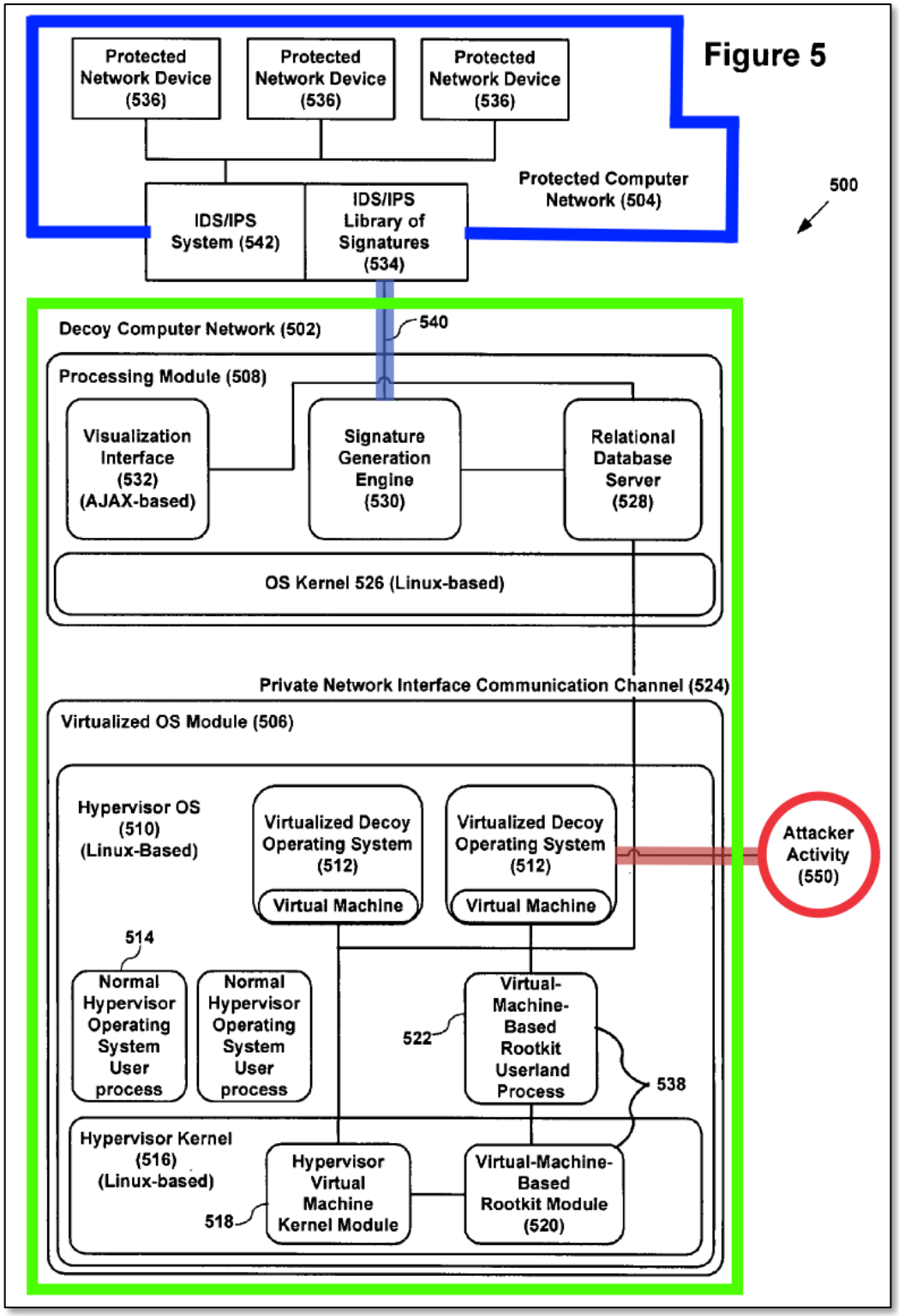
105. *Farley* is also reasonably pertinent to certain problems addressed by the '872 Patent such as providing a forensic analysis of the impact of an attack on a target device. *Compare Farley*, 3:25-46, *with '872 Patent*, 2:20-31, 10:30-55. Thus, *Farley* is analogous art to the '872 Patent.

X. GROUND 1: *CAPALIK* RENDERS CLAIMS 1-6, 12, 15, 20-21 OBVIOUS

A. Claim [1(Pre)]: "A computer implemented method of identifying unauthorized activities on a first computer system attached to a computer network, wherein the first computer system comprises one or more processors and memory, the method comprising:"

106. To the extent the Preamble is limiting, *Capalik* teaches it. *Capalik* describes "a system for analyzing and preventing unauthorized intrusion into a computer network" as shown in FIG. 5. *Capalik*, [0037]; *id.* ("This embodiment is directed to an improved method and system for analyzing unauthorized intrusion into a decoy computer network, the analysis of which is used to prevent unauthorized

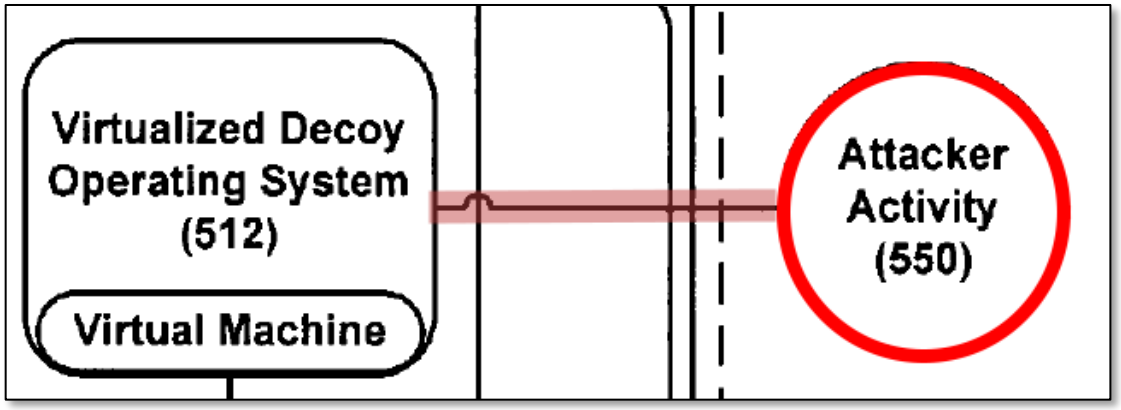
access into a protected computer network.”). Decoy computer network 502 (annotated green) comprises multiple modules that identify unauthorized activities on a decoy computer system:



Capalik, Fig. 5 (annotated); [0038] ("The decoy computer network 502 includes a virtualized operating system module 506 for monitoring the decoy network 502, and

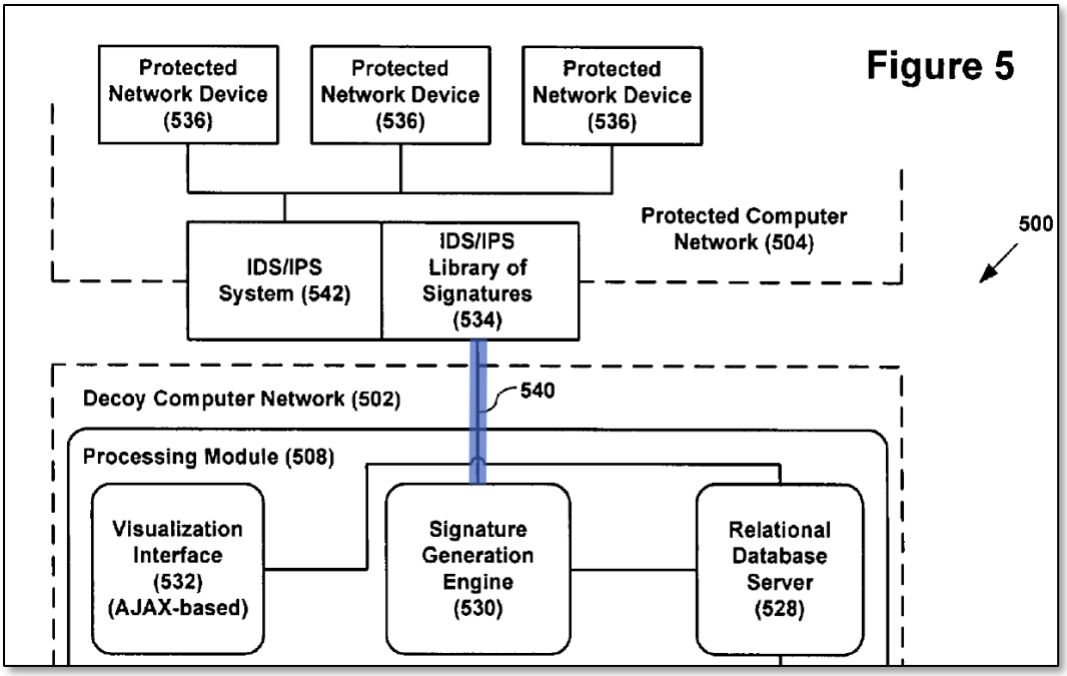
a processing module 508 for obtaining, analyzing, and responding to exploits.”). *Capalik* teaches “[t]hese modules may be hosted on the same computing device or on separate computing devices.” *Capalik*, [0039]. To simplify the discussion, Petitioner focuses on an implementation where the entirety of decoy computer network 502 is implemented on a single computing device and will refer to this device and the modules therein as *Capalik's* “Decoy Computer 502.”

107. *Capalik's* Decoy Computer 502 is attached to a computer network. Namely, *Capalik* teaches “attacker activity 550 is directed at the [D]ecoy [C]omputer [] 502 through one or more ports of each of the virtualized decoy operating systems 512 that are left open as a gateway for attacker activity 550.” *Capalik*, [0044]. This “attacker activity 550” is annotated red in Fig. 5 above. A POSITA would have understood that ports left open on Decoy Computer 502 through which attacker activity can be directed is a reference to network connectivity, facilitating attacker traffic to/from Decoy Computer 502. For example, an attacker would have been able to utilize the network connection such as open ports or sockets to attack virtualized decoy OS 512, which is part of Decoy Computer 502. The annotated excerpt of *Capalik's* Fig. 5 below depicts attacker activity 550 (red circle) and the network connection (light red) used to attack Decoy Computer 502:



Capalik, Fig. 5 (excerpt, annotated). Thus, while *Capalik* does not expressly label the network connection above, a POSITA would have understood that the line (annotated light red) indicates a network connection by which an attacker can attack Decoy Computer 502. Accordingly, a POSITA would have understood Decoy Computer 502 is a first computer system ("computer system") attached to a computer network, as claimed.

108. Additionally, *Capalik* teaches Decoy Computer 502 is attached to "protected computer network 504," annotated blue in Fig. 5 above, via connection 540. *Capalik*, [0054], [0038], Fig. 5. Specifically, annotated Fig. 5 below depicts a second network connection 540, which is a network connection between Decoy Computer 502 and the one or more computers of Protected Computer Network 504.



Capalik, Fig. 5 (annotated, excerpt). *Capalik* teaches second network connection 540 (annotated blue) is "used by the processing module 508 to directly interact with the IDS/IPS system 542." *Capalik*, [0054], [0049] ("When an attacker connects to a network port and begins interacting with a virtualized decoy operating system 512, the introspection module 538 monitors and captures information from the connection[.]").

109. *Capalik's* Decoy Computer 502 includes "one or more processors, input/output devices, communication circuitry, power sources, memory (both physical, e.g., RAM, and non-volatile/non-transitory memory including disks, e.g., hard disk drives), and any other physical hardware necessary for hosting and running the aforementioned modules." *Capalik*, [0039]; *see also Capalik*, claim 21 ("The system of claim 13, **wherein the virtualized operating system module and the**

processing module are contained in memory on the same or separate **computing devices that each includes a processor**.”), claim 22 (“A computing device configured for analyzing unauthorized intrusion into a computer network, **the device comprising: a processor; and memory coupled to the processor[.]**”). Where decoy computer network 502 and all its modules are implemented on the same computing device (i.e., Decoy Computer 502), as is the focus of the Proposed Grounds, a POSITA would have understood that virtualized operating system module 506 and processing module 508—within Decoy Computer 502—run on a computing platform that includes at least a processor and memory, as claimed.

110. Specifically, *Capalik* teaches using a single computing device (i.e., Decoy Computer 502), which includes a processor and memory, and the memory stores modules and “comprises procedures for” the virtualized decoy operating system running on a hypervisor operating system hosted on a “decoy network device” (i.e., Decoy Computer 502). *Capalik*, Claim 22.

B. Claim [1(a)]: “monitoring activity on the first computer system;”

111. Proposing the same structure and using the same terminology as the ’872 Patent, *Capalik* teaches monitoring activity on the computer system using a

virtual machine monitor (i.e., Hypervisor module) that monitors activity on multiple virtual machines within the first computer system.[6]

1. *'872 Patent's* VM and VMM

112. Figure 2 of the '872 Patent depicts an exemplary decoy network device 106 including Virtual Machines (VMs) (annotated pink), and a Virtual Machine Monitor (VMM) (annotated orange). The VMs and VMM are contained in a Hypervisor OS block (annotated green) within the decoy computer, and an

---

[6] Consistent with the '872 Patent's focus on monitoring virtual machines (i.e., decoy devices), the Proposed Grounds focus on *Capalik's* process of monitoring virtual machines within Decoy Computer 502. Although the '872 Patent contemplates *the possibility* of employing its methods on non-decoy systems, all detailed embodiments target decoy device monitoring. *See, e.g., '872 Patent*, 5:25-30 ("Although the methods described herein are described with respect to monitoring the decoy network device 106, . . .similar methods ***may be used*** for monitoring one or more non-decoy systems[.]"). The patent also teaches that virtual machines are an example of the described "computer system." *Id.*, 11:15-18 ("FIGS. 3A-3D and 4A-4E illustrate exemplary network diagrams representing unauthorized activity on a ***computer system*** (***e.g.***, the ***virtual machine*** 113-1)[.]"). Because the disclosed embodiments describe monitoring virtual machine "computer systems," the claims cannot be construed such that the claimed "computer system" excludes virtual machine-based monitors. Indeed, a POSITA would understand that, while the '872 Patent claims do not *require* virtual machines, they certainly *capture* virtual machines.

introspection kernel (annotated blue) is interposed between the VMM and VMs, as illustrated below:

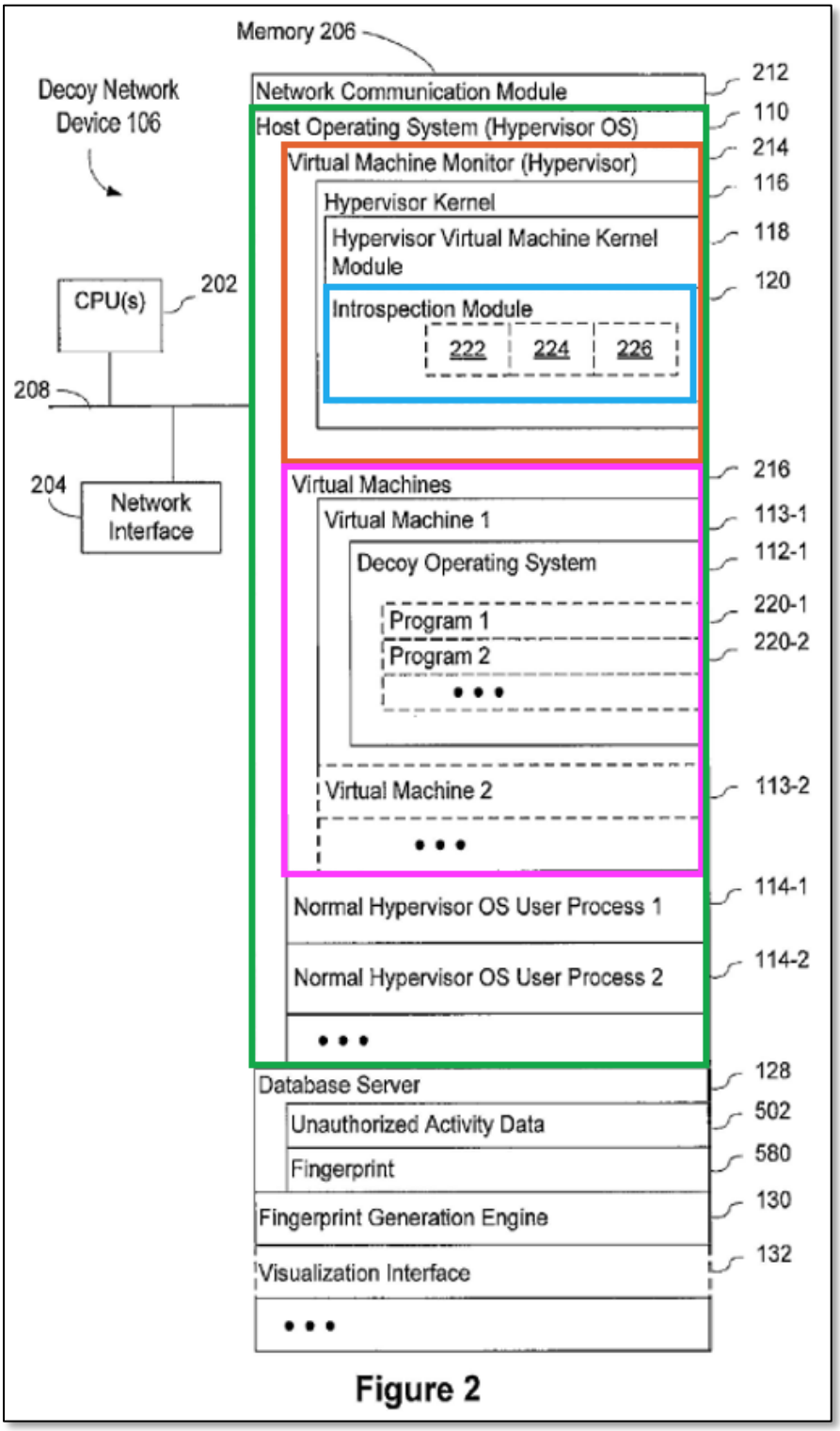
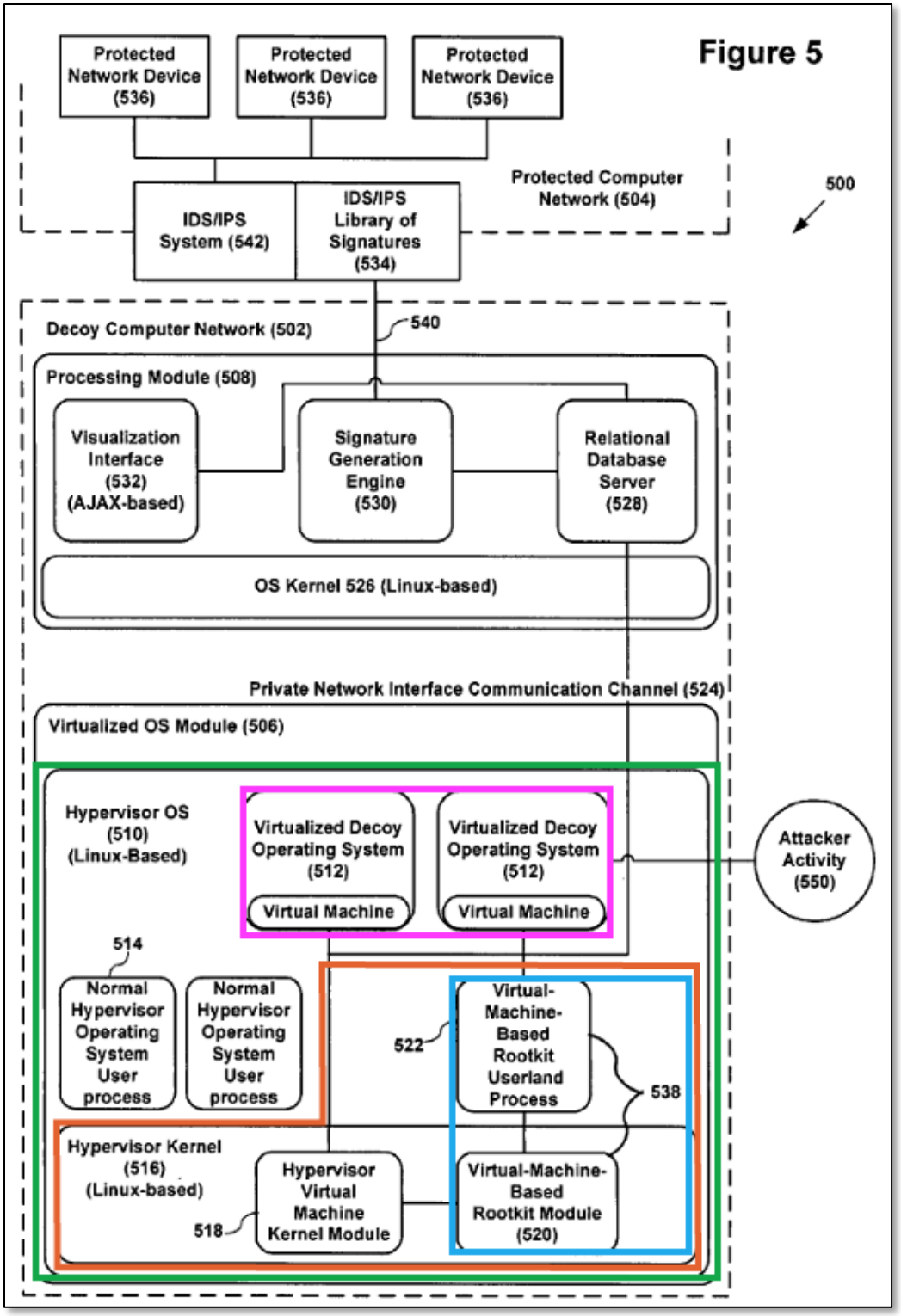


Figure 2

'872 Patent, Fig. 2 (annotated). The '872 Patent explains that the "virtual machine monitor 214 includes a hypervisor kernel 116 that resides in physical memory and provides the basic services to the virtual machine monitor 214." *Id.,* 6:35-37. The introspection kernel "performs introspection (e.g., monitoring) into the physical memory segments assigned to each of the [VM operating systems]." *Id.,* 6:46-52.

2. *Capalik's* VM and VMM

113. *Capalik's* Fig. 5 below depicts the same arrangement, using the same terminology:



Capalik, Fig. 5 (annotated).

114. Like the ’872 Patent, *Capalik* describes Hypervisor OS 510 (annotated green), “also known as a virtual machine monitor operating system[.]” *Capalik*, [0040]. Within Hypervisor OS 510, multiple Virtual Machines (annotated pink) exist in the form of “virtualized operating systems 512 [and] act as decoy operating systems to attract attacker activity 550.” *Id.* Hypervisor kernel 516 (annotated orange) “is that part of the hypervisor operating system 510 that resides in physical memory at all times and provides the basic services to the hypervisor operating system 510.” *Capalik*, [0041]. “[T]he virtual machine-based rootkit module 520 and its associated userland processes 522 constitute an ***introspection module 538*** [(annotated blue)] (also known as a virtual machine-based memory introspection analysis tool) that ***monitors and introspects into the virtualized decoy operating systems memory segments***.” *Capalik*, [0047].

115. In other words, with almost verbatim terminology and functionality, *Capalik* describes the same VM and VMM system as described and claimed in the ’872 Patent. Both the ’872 Patent and *Capalik* use the term “Hypervisor” interchangeably with virtual machine monitor. Compare *’872 Patent*, 6:26 (“virtual machine monitor 214 (also called a hypervisor)”), *with Capalik*, [0040] (“hypervisor operating system 510 (also known as a virtual machine monitor operating system)”). Indeed, in both examples, a VMM is referred to expressly as a hypervisor (or vice versa). A POSITA would have further understood that other prior art references

referred to hypervisor and VMM synonymously, as I discussed above. *See* Section V.C (discussing *Colbert*). And, as depicted above, both deploy "hypervisor kernels" to provide the basic VMM services, including "introspection modules" to monitor the VM memory segments.

116. *Capalik* teaches that Decoy Computer 502's VMs and VMM are stored in memory. Namely, *Capalik* teaches that module 506—the Virtualized OS Module that includes the depicted VMs and VMM—is stored in memory and that the module is "present in physical memory [RAM] once the system has been booted and is operational." *Capalik*, [0038]-[0039].

117. Accordingly, for the reasons discussed above, a POSITA would have understood that *Capalik's* Decoy Computer 502 includes a VM and a VMM (also referred to as a hypervisor) that supervises and monitors activity on the first computer system.

C. Claim [1(b)]: "identifying a plurality of activities being performed at the first computer system, wherein each of the activities includes an activity source, an activity target, and an association between the activity source and the activity target;"

118. As discussed above, an ***association*** can be any event or operation, including calling, executing, modifying, or accessing operation, such as system calls and library calls between an activity source and an activity target. *See* Section III.D.

119. *Capalik* teaches identifying a plurality of activities being performed at the computer system, wherein each of the activities includes an activity source, an

activity target, and an association, as discussed below. For example, a POSITA would have understood that *Capalik* teaches identifying a plurality of activities wherein each activity includes an activity source (e.g., data packet stream), an activity target (e.g., socket receiving the data packet stream), and an association between the activity source and activity target (e.g., receiving the data packet stream at the socket).

1. *Capalik's* plurality of activities performed at the first computer system

120. *Capalik* discloses that when an attacker connects to a network port and begins interacting with virtualized decoy operating system 512, *Capalik's* introspection module 538, "monitors and captures information from the connection, including port numbers, data streams, file uploads, keystrokes, ASCII or binary files, malicious payloads, memory manipulation attempts, and any other data transfers or malicious attempts." *Capalik*, [0049]. The captured information is then (1) sent to processing module 508, (2) stored in relational database server 528, and (3) used to generate an attack report. *Capalik*, [0050], [0052], [0055]. *Capalik* teaches the attack report includes "information about which sockets were accessed by the attack 550, what happened at a particular socket, the keystrokes entered or data transferred, what files were transferred, how the attack 550 was run, what happened on the virtualized operating system module 506, and how the virtualized decoy operating systems 512 [] and any related network services were affected." *Capalik*, [0056]-[0057].

121. *Capalik* teaches the attack may contain one or more data transfers or keystrokes (*Capalik*, [0055]), and the report is "analyzed for known attack patterns as well as non-standard patterns, such as repeating binary patterns, keystroke patterns, downloaded daemons, or errors (such as buffer overflow attempts, malicious payloads attempting to execute arbitrary code on the system, memory overwriting attempts, stack attacks, and heap attacks)." *Capalik*, [0057]. Additionally, *Capalik* teaches that attack 550 may be categorized and analyzed by observing operations performed on decoy operating system(s) 512 "to determine, for example, how an attack gained control of decoy operating system(s) 512. Any method of analyzing the incoming data such as binary matching, neural-network matching, keyword matching, or any other method of matching attack identifying information is in keeping with the spirit of the invention." *Capalik*, [0057]. Thus, a POSITA would have understood that *Capalik* teaches identifying a plurality of activities being performed at the computer system.

2. *Capalik's* activity sources and targets

122. The *'872 Patent* discloses activity sources include access to a particular file, an instruction, a message, data stream, or network port. *'872 Patent*, 13:21-28. *Capalik* teaches identifying at least these activity sources.

123. As discussed above, *Capalik's* introspection module 538, "monitors and captures information from the connection, including port numbers, **data**

streams, **file uploads**, keystrokes, ASCII or binary files, **malicious payloads**, memory manipulation attempts, and **any other data transfers or malicious attempts**." *Capalik*, [0049]. *Capalik* expressly teaches the aforementioned examples of sources, some of which are the same as those described in the '872 Patent. Accordingly, a POSITA would have understood *Capalik's* activity sources included, for example, ports, files, instructions, data streams, and any other data transfers to the decoy computer.

124. The *'872 Patent* discloses activity targets include a file, an instruction, a data stream, and a network port. *'872 Patent*, 13:29-33. *Capalik* teaches identifying at least these activity targets.

125. *Capalik* teaches monitoring processes following receipt of files, malicious payloads, or other data transfers. For example, *Capalik* teaches that attack 550 may be categorized and analyzed by observing operations performed on decoy operating system(s) 512 "to determine, for example, how an attack gained control of decoy operating system(s) 512." *Capalik*, [0055]-[0057]. In one example, a target may be a socket or port of *Capalik's* decoy computer that receives a data transfer from an attacker. *Capalik*, [0056]-[0057], [0059], [0044]-[0045], [0049]. For example, *Capalik* teaches an attack report includes "information about which sockets were accessed by the attack 550, what happened at a particular socket …" and the "operations performed on the decoy operating system(s) 512" as well as

determining when "data is found to be transferred through the socket." *Capalik*, [0056]-[0057], [0059]. Further, *Capalik* teaches monitoring and reporting the series of actions that occur after receiving data, for example. *Capalik*, [0056]-[0057].

126. A POSITA would have understood that such monitoring would have included the "path" or series of steps including and following the receipt of malicious data (or other activity source). In particular, *Capalik* teaches monitoring what actions a source performs on a target to determine how an attack gains control of Decoy Computer 502. As part of its attack analysis, *Capalik* determines identifying what target (e.g., socket) was accessed, what occurred at the socket (e.g., data downloaded), and what other subsequent actions occurred on Decoy Computer 502 for the attack. *Capalik*, [0056]-[0057]. A POSITA would have understood that *Capalik's* activity targets, then, are processes, files, sockets, ports, or other "recipients" of some action by an activity source, as discussed above. Further, some of *Capalik's* activity targets are the same as those described in the '872 Patent, which were listed above. *Capalik*, [0055]-[0057]; *'872 Patent*, 13:21-28.

127. Thus, a POSITA would have understood that the "data streams" and "sockets" disclosed in *Capalik* are examples of activity sources and activity targets, respectively. *See '872 Patent*, 13:21-33.

3. *Capalik's* associations

128. A POSITA would have understood that *Capalik* teaches identifying an "action" (i.e., association) between the activity source and the activity target. *See* Section III.D; *Capalik*, [0049] ("When an attacker connects to a network port and begins interacting with a virtualized decoy operating system 512, the introspection module 538 monitors and captures information from the connection, including port numbers[and] data streams[.]"), [0056] (teaching that the report of the attack can include information about "what happened at a particular socket …how the attack 550 was run, what happened on the virtualized operating system module 506, and how the virtualized decoy operating systems 512 …and any related network services were affected"); *compare with '872 Patent*, 13:34-36. A POSITA would thus have understood that *Capalik* monitors and reports on how activity sources (e.g., stream of malicious data) interact with an activity target (e.g., socket, file, process). For example, *Capalik* teaches activity sources may read or write an activity target. In the example of Fig. 3, a source such as Rbot.CBQ copies itself (reads and then writes) to a file "msprexe.exe" in the "System" directory (step 300), and subsequently first (step 301) and second (step 302) registry entries are created by the attack. *Capalik* teaches "[a]ny other changes to the system may be shown, as part of the attack-identifying information[.]" *Capalik*, [0035].

Figure 3

When first run W32/Rbot-CBQ copies itself to <System>\msprexe.exe. ← 300

The following registry entries are created to run msprexe.exe on startup:

HKCU\Software\Microsoft\Windows\CurrentVersion\Run ← 301
Windows Management System
msprexe.exe

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run ← 302
Windows Management System
msprexe.exe

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices
Windows Management System
msprexe.exe

Registry entries are set as follows:

HKLM\SOFTWARE\Microsoft\Ole
EnableDCOM
N

HKLM\SYSTEM\CurrentControlSet\Control\Lsa
restrictanonymous
1

Capalik, Fig. 3.

129. Because *Capalik* monitors and reports the full "path" or series of actions between a plurality of activity sources and activity targets, a POSITA would have understood that *Capalik* teaches identifying associations between each set of the plurality of activity sources and activity targets. In particular, the full "path" or

series of actions between a plurality of activity sources and activity targets would have included actions performed by activity sources on activity targets, such as reading or writing data. In this way, the association between a source and target may be a write action or read action, for example.

D. Claim [1(c)]: "storing in the memory a data structure that identifies the activity sources, the activity targets, and the associations for the plurality of activities; and"

130. As discussed above, *Capalik* teaches a computer system comprising a memory. *See* Claim [1(Pre)]. Also discussed above, *Capalik* teaches identifying activity sources, activity targets, and associations. *See* Claim [1(b)]. As discussed below, *Capalik* teaches storing ***in the memory*** of the computer system, according to the format specified by a data structure, information that identifies the activity sources, activity targets, and corresponding associations.

131. *Capalik* discloses storing "attack information" in relational database server 528 (shown in blue) in Decoy Computer 502, as depicted in Figure 5:

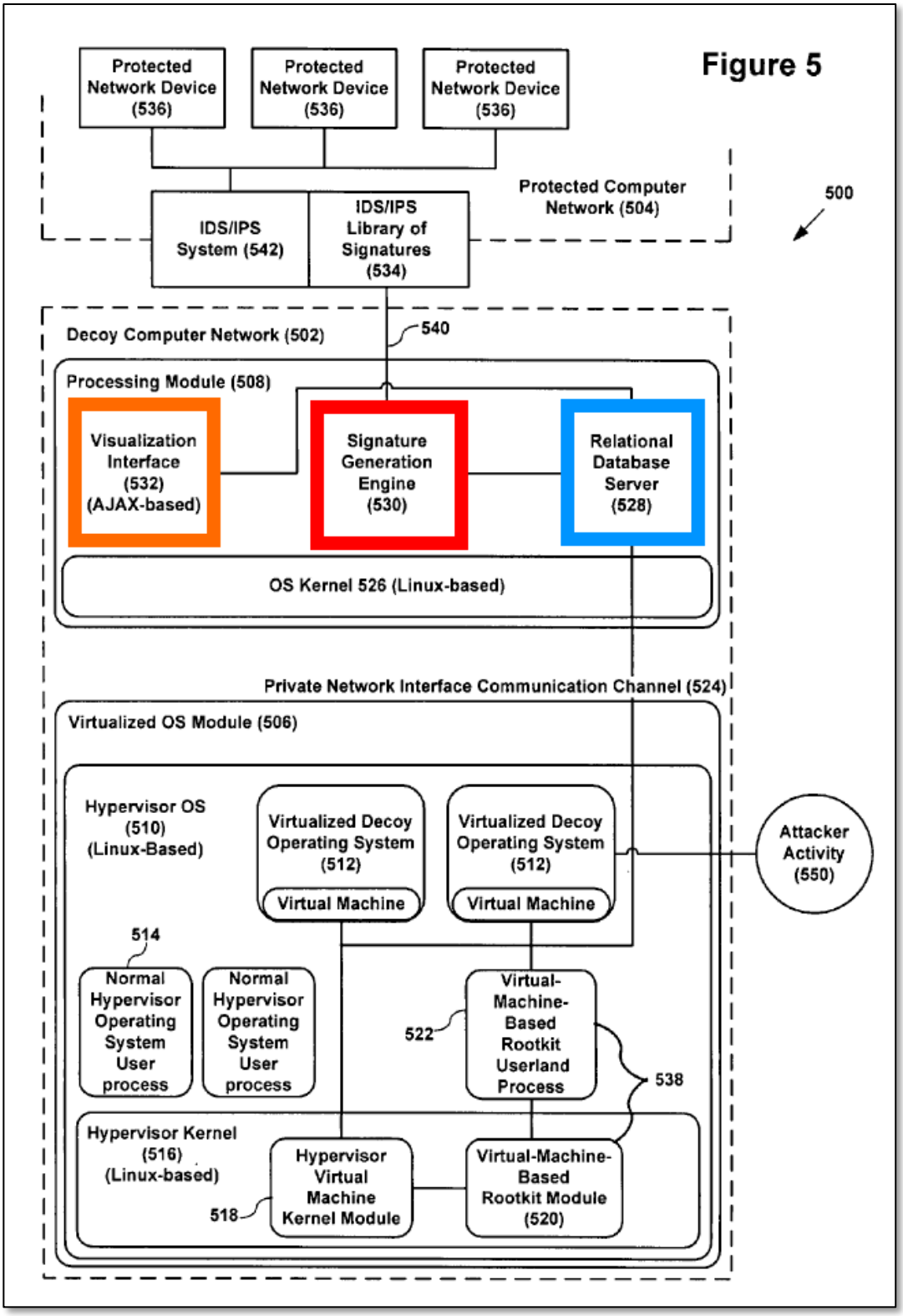


Figure 5
Protected Network Device (536)
Protected Network Device (536)
Protected Network Device (536)
Protected Computer Network (504)
500
IDS/IPS System (542)
IDS/IPS Library of Signatures (534)
Decoy Computer Network (502)
540
Processing Module (508)
Visualization Interface (532) (AJAX-based)
Signature Generation Engine (530)
Relational Database Server (528)
OS Kernel 526 (Linux-based)
Private Network Interface Communication Channel (524)
Virtualized OS Module (506)
Hypervisor OS (510) (Linux-Based)
Virtualized Decoy Operating System (512)
Virtual Machine
Virtualized Decoy Operating System (512)
Virtual Machine
Attacker Activity (550)
514
Normal Hypervisor Operating System User process
Normal Hypervisor Operating System User process
522
Virtual-Machine-Based Rootkit Userland Process
538
Hypervisor Kernel (516) (Linux-based)
518
Hypervisor Virtual Machine Kernel Module
Virtual-Machine-Based Rootkit Module (520)

Capalik, Fig. 5 (annotated). *Capalik* teaches virtualized operating system module 506 monitors decoy network 502, and processing module 508 obtains, analyzes, and responds to malicious exploits. *Capalik*, [0038]. Processing module 508 comprises Relational Database Server 528 (annotated blue), which stores "attack information" such as forensic data for later analysis. *Capalik*, [0052]. *see also*, *id.* [0048] ("the introspection module 538 examines the contents of the decoy operating systems' 512 memory segments in **an instrumented context that generates and retrieves forensic data** for analysis by the processing module 508."). "The signature generation engine 530 [annotated red] then uses this forensic data to generate a signature of the attack." *Capalik*, [0052]. *Capalik's* Visualization Interface 532 (annotated orange) which displays a report of the attack, may include information on the location of the instigator of the attack 550, and may be used to analyze, configure, and automate the system's response to attack activity 550. *Capalik*, [0056].

132. The '872 Patent explains that its method includes storing the activity sources, activity targets, and associations locally in the memory of the decoy network device 106, as depicted in Fig. 2:

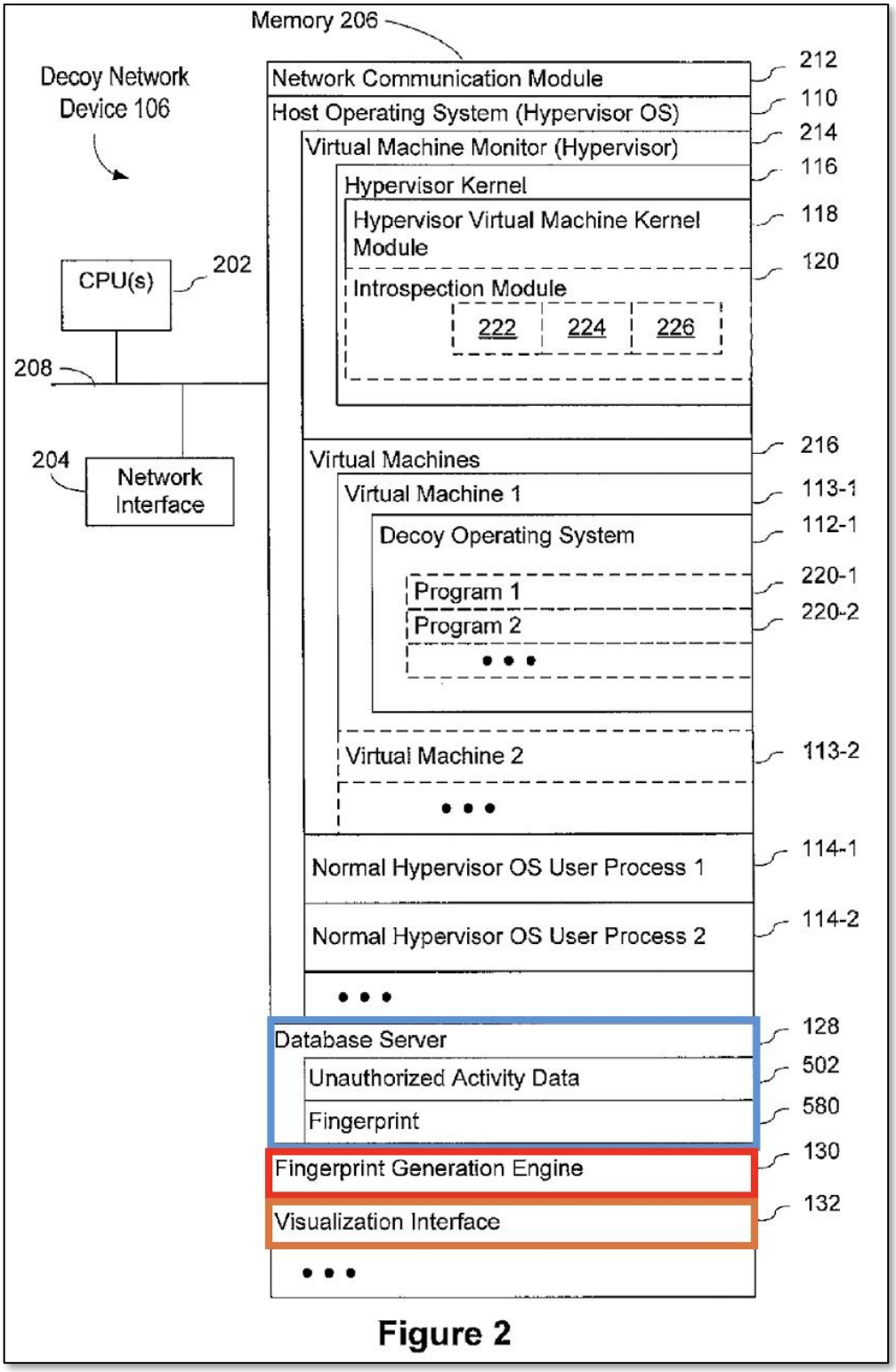


Figure 2

'872 Patent, Fig. 2 (annotated). As depicted in Fig. 2 above, decoy network device 106 includes memory 206 which stores one or more of the following programs: a

"database server 128 [shown in blue], which stores data (e.g., unauthorized activity data 502 and/or fingerprint 580)." *Id.*, 6:10-11. A "fingerprint generation engine 130 [shown in red], which generates the fingerprint data 580 from the unauthorized activity data 502)." *Id.*, 6:12-14. And a "visualization interface 132 [shown in orange], which prepares data (e.g., the fingerprint or unauthorized activities on the computer system) for display (e.g., on a GUI of a client 101)." *Id.*, 6:15-17. As such, there are both structural and functional similarities between the "storing" as taught by *Capalik* and the '872 patent.

133. As explained in Claim [1(a)], the modules within *Capalik's* Decoy Computer 502 utilize memory (e.g., volatile memory (e.g., RAM) and non-volatile memory (e.g., hard disk drive)) to accomplish their respective tasks. *Capalik*, [0039]. A POSITA would have understood that "attack information" stored in Relational Database Server 528 uses system memory (e.g., RAM, hard disk drive), as claimed. Specifically, a POSITA would have understood that memory and storage such as RAM and hard disk drives were commonly used for storing information either for the short term (e.g., volatile, not maintained when system power is turned off) or long term in a hard disk (i.e., non-volatile storage, such as magnetic storage, which maintains information even when not receiving power). Because *Capalik* teaches two specific types of memory for Decoy Computer 502, a POSITA would have understood that Relational Database Server 528 (which is part of Decoy Computer

502) would have used the available memory of Decoy Computer 502 (either volatile and/or non-volatile/non-transitory memory) to store attack information.

134. A POSITA would have understood or found obvious that *Capalik's* "forensic data" is stored in a data structure, as claimed, at least because the stored data is used by "[t]he signature generation engine 530 [annotated red] . . . to generate a signature of the attack." *Capalik*, [0052]; *compare with '872 Patent,* 10:33-36. Storing data such as forensic data in data structures was known in the art. For example, *Bhattacharya* teaches an event log storing information in an "event message" including source address, source port, destination address, destination port, IP protocol, timestamp, event ID, and event type. *Bhattacharya*, 3:51-4:60. *Bhattacharya's* Fig. 2 below depicts a series of such event messages that are stored in a network security monitor's event log:

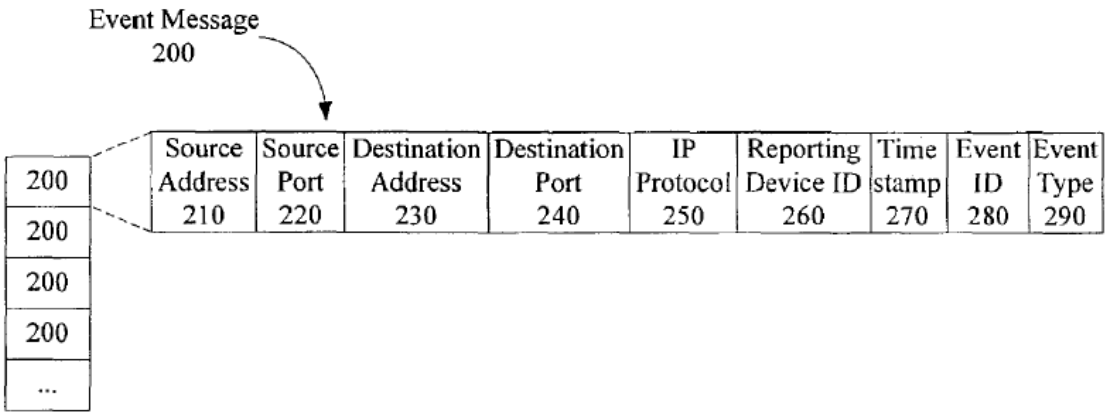


Fig. 2

Bhattacharya, Fig. 2. *See also Farley*, Figs. 5B-5C (depicting a data structure containing data corresponding to a raw event including source IP address, destination IP address, event type, time stamp, etc.).

135. *Capalik's* signature generation engine 530 would use a data structure to effectively create a signature from the stored forensic data. Indeed, the '872 Patent claims do not impose any specific structure of the stored data, generically recited as "data structure." Given the breadth of this element and the fact that *Capalik's* system processes the stored data to generate a signature of the attack, a POSITA would have understood that *Capalik* teaches or suggests storing its forensic data in a "data structure."

136. Moreover, *Capalik* teaches "processing module 508 generates a report of the attack that includes attack-identifying information." *Capalik*, [0055]. A "report of the attack is written and then displayed via a visualization interface 532 [annotated orange]" *Capalik*, [0056]. As explained above at Claim [1(b)], *Capalik's* attack report includes activity sources and targets, and the corresponding associations. Accordingly, a POSITA would have understood that the "attack information" or "attack-identifying information" stored in relational database server 528 and used to create the attack report includes activity sources, targets, and associations. *See* Claim [1(b)].

E. Claim [1(d)]: "transmitting to one or more computer systems other than the first computer system information identifying one or more of the activity sources, the activity targets, and the associations for preventing future attacks, to the one or more computer systems, associated with the one or more of the activity sources, the activity targets, and the associations"

137. As discussed above, *Capalik* teaches storing in the memory of the computer system data in a data structure, where the data identifies the activity sources, activity targets, and corresponding associations. *See* Claim [1(c)]. Satisfying this limitation, *Capalik* teaches creating a signature (i.e. fingerprint) from the stored forensic data, which contains information identifying one or more of the activity sources, targets and associations.

1. *'872 Patent's* Description of Fingerprint Generation

138. The '872 Patent describes generating a fingerprint[7] indicative of attack activity that "can be used to prevent future attacks[,] by modifying the activity targets[,]" such as "blocking a particular port[.]"[8] *'872 Patent*, 13:65-14:16; *see also id.*, 14:42-50 (teaching that fingerprints can also prevent attacks by "blocking activity sources" that match an attack on the virtual machine). The fingerprint can include "one or more of" an activity source, activity target, and other data descriptive of an identified attack. *Id.*, 14:1-4. The patent is clear that fingerprints can include both source and target data or even just source or target data. *Id.*, 14:17-53 ("In some embodiments, the fingerprint [] includes activity targets [] in lieu of or in addition to, the activity sources[.]").

2. *Capalik's* Signature/Fingerprint Generation

139. Like the '872 Patent, *Capalik* teaches a decoy system (Decoy Computer 502) comprising Relational Database Server 528, which stores "attack information" using a forensic data structure. *Capalik*, [0052]; *see* Claim [1(c)]. Also, like the '872

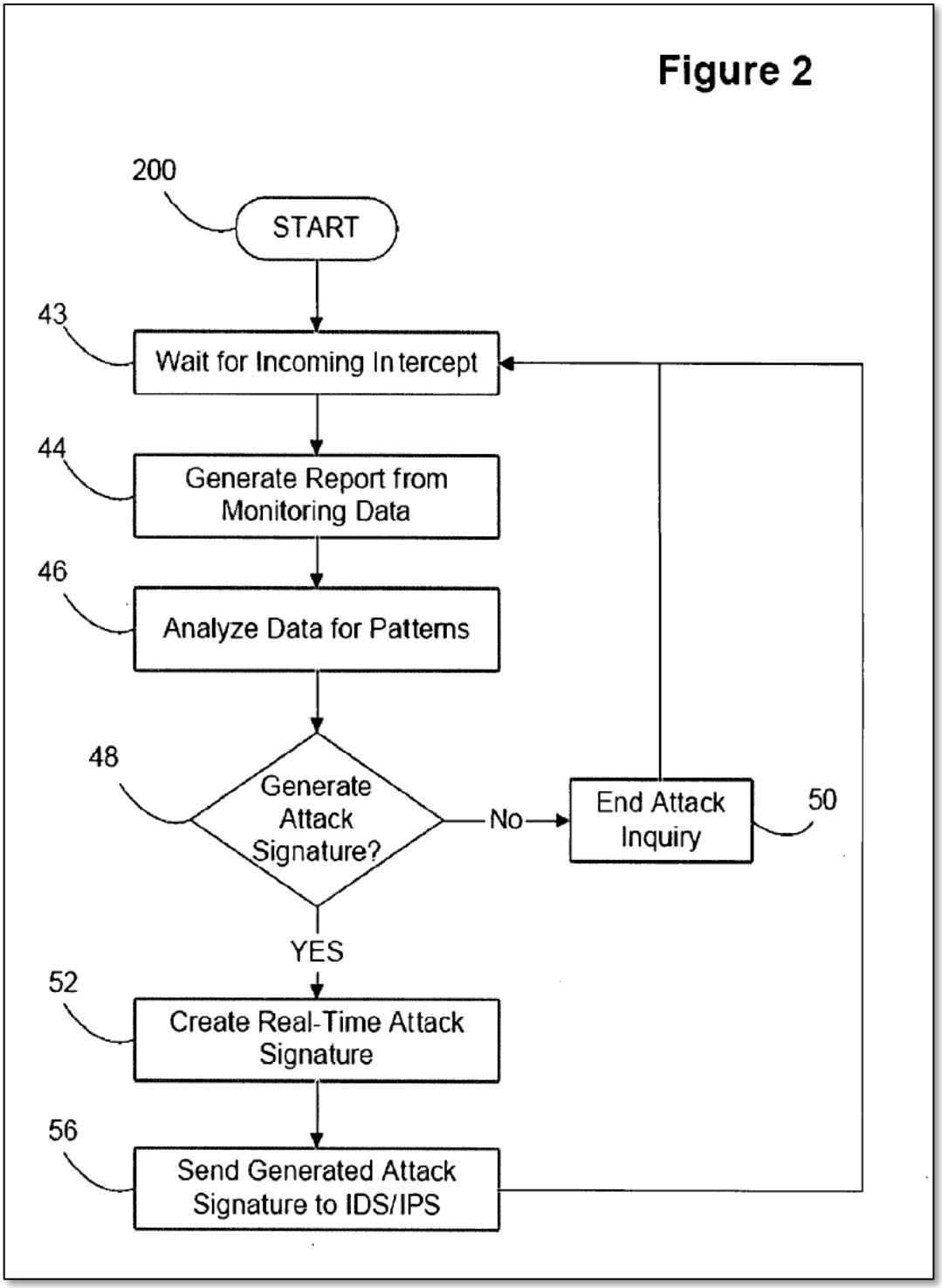
---

[7] The '872 Patent also refers to a fingerprint as a signature. *'872 Patent*, 10:33-55.

[8] Although *Capalik* and the '872 Patent include many similar examples, both expressly teach generating a fingerprint/signature of an attack that identifies the port (or socket) targeted by the attack. I have highlighted this parallel teaching merely as an illustrative example, but my opinions are not limited to this teaching.

Patent, *Capalik* uses a signature-generation engine to create a "signature of the attack" from the stored forensic data in a relational database. *Compare Capalik*, [0052], *with '872 Patent*, 6:10-14. A POSITA would have understood that *Capalik's* signature is a "fingerprint." The '872 Patent describes signature and fingerprint as synonymous. *'872 Patent*, 10:49-50 ("The signature (also called a fingerprint)[.]"), 10:30-61.

140. *Capalik* discloses decoy system processing module 508 of Decoy Computer 502 generates an attack report containing attack-identifying information, analyzes the information for attack patterns, and generates an attack signature. *Capalik*, [0055]. In detail, *Capalik* teaches a method depicted in Fig. 2 that applies to Decoy Computer 502 (depicted in Fig. 5). *Capalik*, [0037]. A POSITA would have understood that Fig. 2 is a flowchart of steps that are performed by components of Decoy Computer 502 (as depicted in Fig. 5) discussed in more detail below. Regarding Fig. 2, *Capalik* teaches waiting for an incoming intercept/attack in step 43 and then generating a report of the attack from monitoring data in step 44. *Capalik*, [0055]-[0056]. After an attack report is generated, a signature can be generated in step 52. *Capalik*, [0051]-[0052]. Fig. 2 below depicts this process:

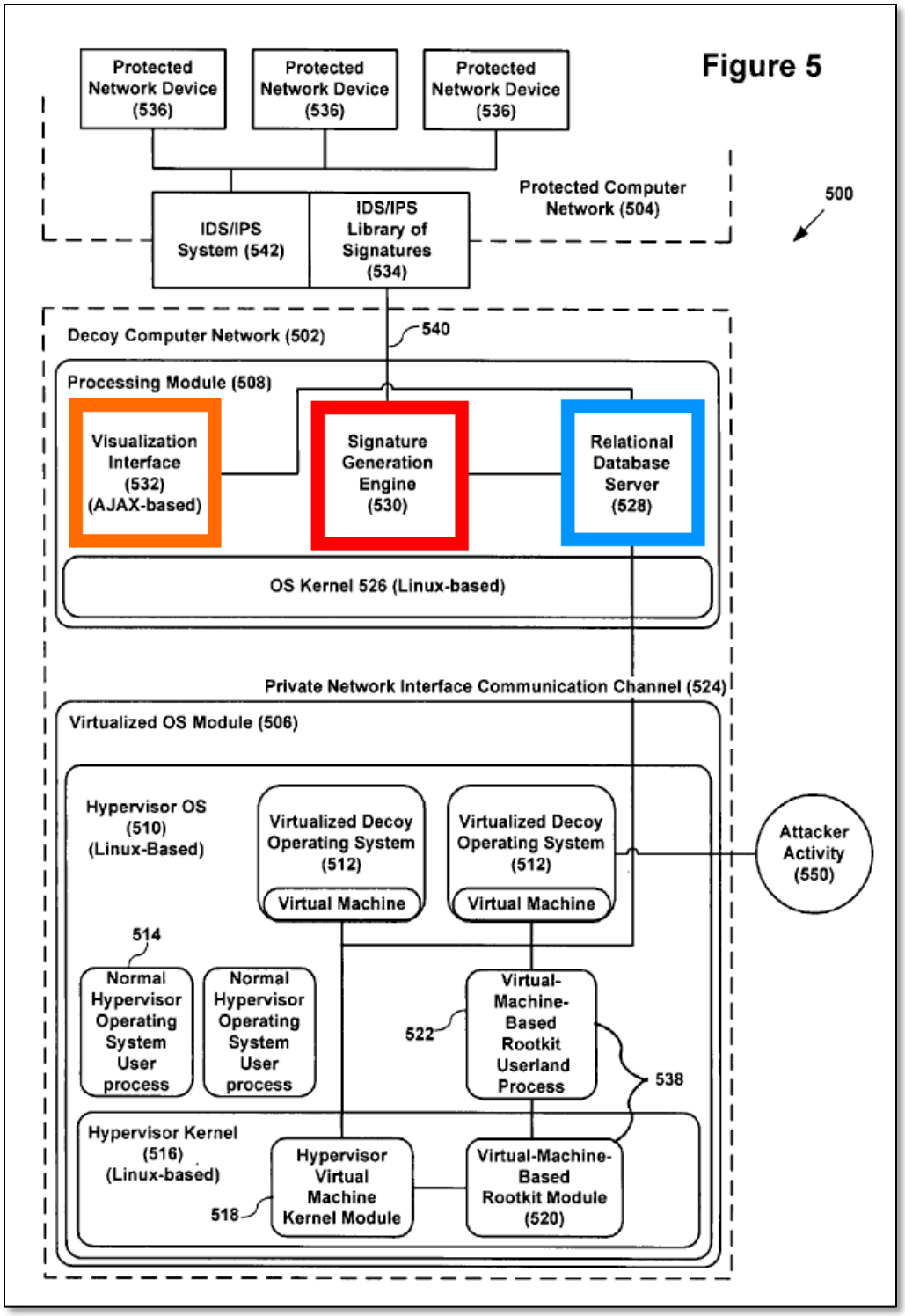


Capalik, Fig. 2. Regarding the attack report of Fig. 3 and step 44, *Capalik* teaches presenting the attack report to an admin. *Capalik's* visualization interface 532

(annotated orange in Fig. 5 below) "displays a report of the attack, may include information on the location of the instigator of the attack 550, and may be used to analyze, configure, and automate the system's response to attack activity 550." *Capalik*, [0055]-[0056]. A POSITA would have understood that visualization interface 532 thus may be used to automate the process of signature generation, for example. *Capalik*, [0056], [0052] ("The entire process from attack detection through signature generation may occur automatically[.]"). Alternatively, a POSITA would have understood that an admin would manually review a report and choose to generate a signature. *Capalik*, [0055], [0058]-[0059].

141. In more detail regarding the components of Fig. 5 that perform the steps of the flowchart depicted in Fig. 2, Decoy Computer 502 includes processing module 508, which also includes signature-generation engine 530. In particular, *Capalik's* signature-generation engine 530 generates signatures of attacks. *Capalik*, [0051] ("The processing module 508 also includes a database, such as a relational database server 528, and a signature-generation engine 530."). In more detail, *Capalik* teaches introspection module 538 (part of Hypervisor OS 510 of Decoy Computer 502) captures attack information and then transmits the information from virtualized OS module 506 to processing module 508 via "private network interface communication channel 524," which is a communication channel within Decoy Computer 502. *Capalik*, [0051]-[0052]. After being transmitted, the attack information is stored as

forensic data on relational database server 528 of Decoy Computer 502. *Id.* Signature-generation engine 530 uses this forensic data to generate a signature of the attack. *Id. Capalik's* annotated Fig. 5 below depicts signature-generation engine 530 (red) and relational database server 528 (blue):



Capalik, Fig. 5 (annotated).

142. Like the ’872 Patent, *Capalik*’s “signature of the attack” can include a wide variety of information, including “what happened at a ***particular socket***, the keystrokes entered or data transferred, what files were transferred, how the attack 550 was run, what happened on the virtualized operating system module 506, and how the virtualized decoy operating systems 512 running on the hypervisor operating system 510 and any related network services were affected.” *Capalik*, [0056]. *Capalik’s* Fig. 4 illustrates an “attack signature[,]” formatted in XML for transmission to an IDS/IPS system:

```
<item type="STRING.TCP">                ← 400
  <struct name="STRING.TCP">
    <var name="version" protected="true">4.0</var>
    <array name="signatures">
      <entry dontDelete="true" nda="false" sfr="85">
        <var name="SigName" default="RBOT.CBQ Worm Activity" protected="true" />
        <var name="SIGID" default="5571" protected="true" />
        <var name="SubSig" default="0" protected="true" />
        <var name="AlarmSeverity" default="high" />
        <var name="Enabled" default="True" />
        <var name="EventAction" default="alarm">alarm|reset</var>
        <var name="SigVersion" default="S185" />
        <var name="SigStringInfo" default="RBOT.CBQ" />
        <var name="AlarmThrottle" default="Summarize" />
        <var name="Direction" protected="true" default="ToService" />
        <var name="MinHits" default="1" />
        <var name="Protocol" default="TCP" />
        <var name="RegexString" protected="true"
default="\xb5\x71\xc4\x43\x1e\x34\xce\x3a\x18\x37\xef\xc3\x22\xa1\x20\x1f\x6c\x10\x8f\x6
8\x3d\xf4\xef\x51\x92\xf9\x4f\xbc\x46\xe9\x05\xdc" />
        <var name="ServicePorts" default="445" />
        <var name="StorageKey" default="STREAM" />
        <var name="SummaryKey" default="Axxx" />
        <var name="ThrottleInterval" default="15" />
      </entry>
    </array>
  </struct>
</item>
```

Capalik, Fig. 4. As illustrated, *Capalik’s* signature includes many details indicative of the activity on a virtual machine, including identifying the specific port on the

virtual machine through which the attack occurred. For example, a POSITA would have understood "service port" 445 refers to a TCP port, which is commonly targeted by malicious attacks. TCP stands for Transmission Control Protocol, which is a commonly used protocol for transmitting information over the Internet. *Capalik* further teaches that "any ports, protocols, severity levels, alarm levels, signature name, or any other quantity, may be utilized to inform an IDS or IPS of an attack signature." *Capalik*, [0062], [0036].

143. Once the signature is created, *Capalik* discloses transmitting its signature in a similar manner as the '872 Patent. The '872 Patent explains that "the decoy network device transmits the fingerprint to the IDS/IPS system 142 to prevent future attacks that comprise the same or similar activities on one or more protected network devices 136." *'872 Patent*, 18:47-51. *Capalik* similarly teaches that once the attack signature is generated, it can be sent "to the intrusion detection system (IDS) or intrusion prevention system (IPS) for the protected network 504." *Capalik*, [0059], Fig. 5; *see also id*., [0052] (teaching that the entire process may occur automatically, without human intervention). The attack signature is then applied to the IDS/IPS library of signatures, updating the library, to prevent an attacker from accessing protected network 504 in a future similar attack. *Capalik*, [0059], *see also id*., [0053] (teaching that "protected computer network 504 includes an IDS/IPS library of signatures 534 and an IDS/IPS system 542 coupled to multiple protected

network devices 536"). A POSITA would have understood that signatures updated in the signature library are also transmitted to the multiple protected network devices 536 coupled to the IDS/IPS system, ensuring those network devices receive protection consistent with the most up to date signatures collected in the library.

144. Accordingly, a POSITA would have understood that *Capalik* transmits the fingerprint, including information identifying one or more of the activity sources, the activity targets, and the associations to prevent future attacks that comprise the same or similar activities as indicated by the fingerprint. A POSITA would have further understood that *Capalik* transmits this information to a computer system other than computer decoy 502 since the signature is sent "through a standard network connection" to IDS/IPS 534 of protected network 504. *Capalik*, [0059], Fig. 5.

F. Claim 2: "The method of claim 1, wherein the monitoring further comprises monitoring all activity on the first computer system."

145. As explained above, *Capalik* teaches monitoring activity on a virtual decoy system. *See* Claim [1(b)]. *Capalik* further teaches monitoring all activity on the VM. *Capalik,* [0059] ("[T]he processing module 508 may generate a signature ***whenever data is found to be transferred through the socket***."), [0049] ("When an attacker connects to a network port and begins interacting with a virtualized decoy operating system 512, the introspection module 538 monitors and captures information from the connection, including port numbers, data streams, file uploads,

keystrokes, ASCII or binary files, malicious payloads, memory manipulation attempts, and any other data transfers or malicious attempts."). A POSITA would have understood *Capalik's* detailed forensic data monitors all activity on the computer system that begins with a connection through a VM network port.

146. Honeypots and decoy systems were known to both monitor all activity at the decoy system as well as assume all activity was unauthorized. As explained above, the point of a honeypot is to act as bait for an attacker. *See* Section V.B. To be an efficient decoy, the system would monitor all activity to prevent any accidental allowance of an attacker through. Similarly, for exhaustive purposes, the honeypot was known to assume all activity at the honeypot was unauthorized. *Id.* By acting as a fake port, any activity that reaches the honeypot is assumed unauthorized since any authorized activity would bypass the honeypot. *Id.* Accordingly, a POSITA would have understood that *Capalik* teaches monitoring all activity on the computer system.

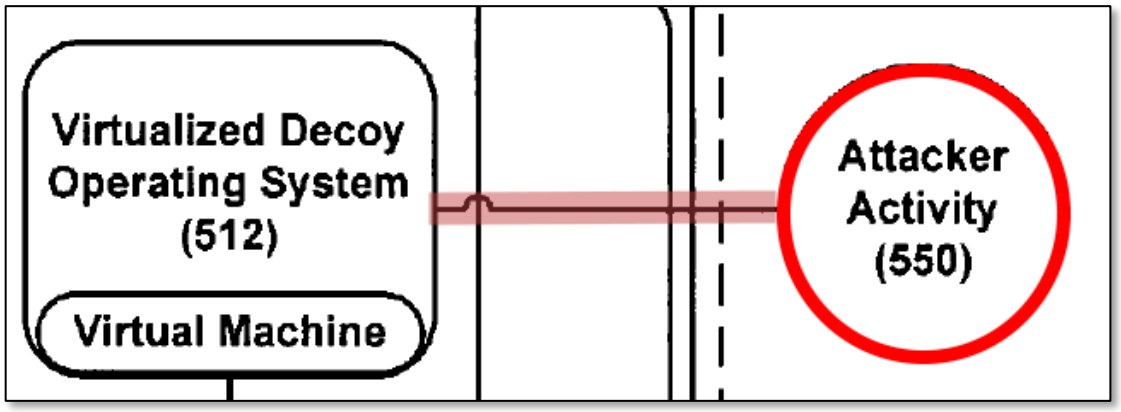
G. Claim 3: "The method of claim 1, further comprising: creating, from the stored information, a fingerprint indicative of the activity on the first computer system."

147. *See* Claim [1(d)].

H. Claim 4: "The method of claim 1, wherein the activities comprise two or more of: a file initiating an execution of an instruction, an instruction reading a file, an instruction writing to a file, and receiving data through the network"

148. *Capalik* teaches identifying a plurality of activities performed in a first computer system, wherein each activity includes an activity source, target, and association. *See* limitation [1(b)]. *Capalik* also teaches a decoy computer (first computer system) attached to a network. *See* limitation [1(Pre)].

149. When an attacker connects to a network port of decoy computer and begins interacting with virtualized decoy OS 512, *Capalik's* system "monitors and captures information from the connection, including port numbers, data streams, file uploads, keystrokes, ASCII or binary files, malicious payloads, memory manipulation attempts, and any other data transfers or malicious attempts." *Capalik*, [0049]. A POSITA would have understood that such attacks would have occurred over a network connection, such as depicted in Fig. 5 and discussed above.



Capalik, Fig. 5 (excerpt, annotated); *see* Claim [1(Pre)]. Thus, a POSITA would have understood that *Capalik* discloses receiving data through network 502.

150. Moreover, *Capalik's* figure 3 illustrates a human readable summary of an example attack, wherein worm W32/RBot-CBQ copies itself (reads and then writes) to a file "msprexe.exe" in the "System" directory (step 300), and subsequently first (step 301) and second (step 302) registry entries are created by the attack.

When first run W32/Rbot-CBQ copies itself to <System>\msprexe.exe. ← 300

The following registry entries are created to run msprexe.exe on startup:

HKCU\Software\Microsoft\Windows\CurrentVersion\Run ← 301
Windows Management System
msprexe.exe

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run ← 302
Windows Management System
msprexe.exe

HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices
Windows Management System
msprexe.exe

Registry entries are set as follows:

HKLM\SOFTWARE\Microsoft\Ole
EnableDCOM
N

HKLM\SYSTEM\CurrentControlSet\Control\Lsa
restrictanonymous
1

Capalik, Fig. 3, [0061]. Based on *Capalik's* disclosures at Fig. 3 and [0061], a POSITA would have understood that Rbot-CBQ initiates execution of both (1) an instruction of reading a file (i.e. Rbot-CBQ reading file "msprexe.exe") and (2) an instruction writing to a file (i.e., Rbot-CBQ writing to file "msprexe.exe"). *Capalik*

teaches attack reports may include information including "one or more data transfers or keystrokes[.]" *Capalik*, [0055]. Indeed, actions such as reading/writing to a file would have been in response to instructions either directly or indirectly from an attacker. When the instructions are executed, actions such as reading and writing would have occurred accordingly.

I. Claim 5: "The method of claim 1, wherein a respective activity target comprises one of: a file, a user, an instruction, a thread, data stream, and network socket connections"

151. *Capalik* teaches activity targets include processes, files, and sockets. *See* Claim [1(b)]. For example, *Capalik* teaches the created attack report includes "which **sockets** were accessed by the attack[.]" *Capalik*, [0056]. A POSITA would have understood that *Capalik's* sockets are "network socket connections" because a socket is a well-known component of a network that is used for connecting devices over the network.

J. Claim 6: "The method of claim 1, wherein a respective activity source comprises one of: a file, a user, an instruction, a thread, data stream, and network socket connections "

152. *Capalik* teaches activity sources include files, malicious payloads, or other data transfers to the decoy computer. *See* Claim [1(b)]; *Capalik*, [0049] (teaching monitoring sources "including **port numbers**, **data streams**, **file uploads**, keystrokes, ASCII or binary files, **malicious payloads**, memory manipulation attempts, and **any other data transfers or malicious attempts**"), [0056]

("information about which sockets were accessed by the attack 550, what happened at a particular socket[.]"), [0059] (teaching sockets). A POSITA would have understood these comprise at least a file, an instruction, data stream, and/or network socket connection. In particular, *Capalik* teaches examples of activity sources including file uploads (file); instructions such as keystrokes, or those contained in files, or malicious payloads; data streams (data stream); and port numbers and sockets (network socket connection.

K. Claim 12: "The method of claim 1, further comprising: identifying activity sources that are affected by unauthorized activities"

153. *Capalik* teaches identifying a plurality of activities being performed on a first computer system, wherein each of the activities includes an activity source, target, and association. *See* limitation [1(b)]. Because the identified plurality of activities includes unauthorized activities, a POSITA would have understood that *Capalik* also teaches identifying activity sources that are affected by unauthorized activities. *See* Claims [1(Pre)], [1(c)].

154. The '872 Patent explains that "[o]nce **the packet stream 304 and the socket 306 are identified as affected by unauthorized activities**, related unauthorized activities are identified." *'872 Patent*, 11:45-47, Fig. 3B. In this example, an activity source, packet stream 304, is identified as affected by unauthorized activity. Accordingly, a POSITA would have understood Claim 12 to include identifying sources coming from outside of the virtual machine (e.g., Virtual

Machine 113-1) as being affected by unauthorized activity. *Capalik* teaches this. For example, *Capalik* teaches Decoy Computer 502 responds "to connection attempts made at network addresses that do not exist[,]" and these "non-existent network addresses are assumed to be malicious[.]" *Capalik*, [0044]. Furthermore, "[w]hen an attacker connects to a network port[,]" the system "monitors and captures information from the connection, including … malicious payloads, memory manipulation attempts, and any other data transfers or malicious attempts." *Capalik*, [0049]. A POSITA would have understood that *Capalik's* assumption of a malicious connection and subsequent monitoring of any data transfer is the same as the '872 Patent's identifying an activity source affected by unauthorized activity, as explained within the '872 Patent. *'872 Patent*, 11:45-47; *compare*, *'872 Patent*, Fig. 3B (showing packet stream 304 coming from outside Virtual Machine 113-1) and *Capalik*, Fig. 5 (showing Attacker Activity 550 coming from outside Decoy Computer 502 and Virtual Machine).

L. Claim 15: "The method of claim 12, wherein the identifying activity sources that are affected by unauthorized activities includes identifying data stream incoming from outside the first computer system."

155. *See* Claim 12. *Capalik* discloses identifying activity sources that are affected by unauthorized activities including identifying data stream incoming from outside the computer system. *Capalik*, [0049] ("When an attacker connects to a network port and begins interacting with a virtualized decoy operating system 512,

the introspection module 538 monitors and captures information from the connection, including port numbers, ***data streams***, file uploads, keystrokes, … and any other data transfers[.]"); Fig. 5 (depicting attacker's data stream incoming from outside computer system 500). For the same reasons stated above at Claim 12, a POSITA would have understood *Capalik* teaches identifying a data stream affected by unauthorized activity (e.g., Attacker Activity 550) incoming from outside the computer system (e.g., outside of Decoy Computer 502).

M. Claim [20(Pre)]: "A first computer system, comprising: one or more processors; and memory storing one or more programs, the one or more programs including instructions for:"

156. See Claim [1(b)]. As discussed above in Claims [1(a)]-[1(d)], *Capalik* teaches the following components: (1) hypervisor OS (comprising an introspection module); (2) relational database server; and (3) fingerprint generation engine. A POSITA would have understood that each of these components comprise one or more programs, respectively including instructions for (1) monitoring activity on the computer system and identifying a plurality of activities being performed at the computer system, where each of the activities includes an activity source, target, and an association; (2) storing in the memory the activity sources, targets, and associations; and (3) creating, from the stored activities, a fingerprint, and transmitting the fingerprint to prevent future similar attacks.

157. In particular, a POSITA would have understood that *Capalik's* modules such as hypervisor OS and components such as the relational database server and fingerprint generation engine would have included instructions for performing their functions. A POSITA would have understood that utilizing programs and/or programming instructions to perform operations on a computer was a well-known concept in computer networking, computer science, and similar fields. Thus, *Capalik's* system, which includes these components, would also have included one or more programs including instructions for performing the claimed steps below. See Claim [20(b)]-[20(d)].

N. Claim [20(a)]: "monitoring activity on the first computer system;"

158. *See* Claim [1(a)].

O. Claim [20(b)]: "identifying a plurality of activities being performed at the first computer system, where each of the activities includes an activity source, an activity target, and an association between the activity source and the activity target;"

159. *See* Claim [1(b)].

P. Claim [20(c)]: "storing in the memory a data structure that identifies the activity sources, the activity targets, and the associations for the plurality of activities; and"

160. *See* Claim [1(c)].

Q. Claim [20(d)]: "transmitting to one or more computer systems other than the first computer system information identifying one or more of the activity sources, the activity targets, and the associations for preventing future attacks, to the one or more computer systems, associated with the one or more of the activity sources, the activity targets, and the associations."

161. *See* Claim [1(d)].

R. Claim [21(Pre)]: "A computer readable storage medium, including one or more programs for execution by one or more processors of a first computer system, the one or more programs including instructions for:"

162. *See supra* Claim [1(Pre)], [20(a)] (mapping one or more programs including instructions for performing the claimed steps, where the mappings for said programs also apply for Claim 20). To the extent the preamble is limiting, a POSITA would have understood that *Capalik's* decoy system discussed in Claim [1(Pre)] includes a computer readable storage medium (e.g., hard disk). As discussed above, a POSITA would have understood that a hard disk is an example of non-transitory/non-volatile memory because hard disks maintain their data even after a computer/system is powered down by utilizing magnetic storage, for example. *See* Claim [1(a)].

S. Claim [21(a)]: "monitoring activity on the first computer system;"

163. *See* Claim [1(a)].

T. Claim [21(b)]: "identifying a plurality of activities being performed at the first computer system, where each of the activities includes an activity source, an activity target, and an association between the activity source and the activity target;"

164. *See* Claim [1(b)].

U. Claim [21(c)]: "storing in memory of the first computer system a data structure that identifies the activity sources, the activity targets, and the associations for the plurality of activities; and

165. *See* Claim [1(c)].

V. Claim [21(d)]: "transmitting to one or more computer systems other than the first computer system information identifying one or more of the activity sources, activity targets, and associations for preventing future attacks, to the one or more computer systems, associated with the one or more of the activity sources, the activity targets, and the associations."

166. *See* Claim [1(d)].

XI. GROUND 2: *CAPALIK* IN VIEW OF *GOEL* RENDERS CLAIMS 1-6, 12, 14-15, 20-21 OBVIOUS

167. As explained in Ground 1, *Capalik* teaches a computer implemented method for identifying a plurality of activities (including activity sources, activity targets, and associations) performed at the computer system, and retaining detailed forensic data that can be used to generate signatures indicative of the activities. *See* Ground 1, Claims 1, 19, 20.

168. To the extent Patent Owner disputes that *Capalik* identifies and stores a source, target, and association for each of the individual activities of an attack, *Goel* demonstrates that it was known in the art to identify and store each of the

individual activities of an attack (including the activity source, target, and association) to generate signatures indicative of the activities for forensic analysis.

A. *Capalik-Goel* teaches identifying and storing "activity sources, activity targets, and associations" as recited in independent Claims 1, 19-20.

169. To avoid repetition, the *Capalik-Goel* combination set forth here highlights *Goel's* role in teaching identifying and storing unauthorized activities wherein each activity includes an activity source, an activity target, and association, as recited in Claims [1(b)], [20(b)], and [21(b)]. Petitioner relies on *Capalik's* teachings as set forth in Ground 1 for the rest of the limitations of these independent claims. *See* Ground 1, Claims [1[Pre]-[1(a)], [1(c)]-[1(d)].

170. *Goel* teaches analyzing intrusions in a computer system using the Forensix system. *Goel*, 1. Forensix monitors and captures all system activity at a "target system" after an intrusion, including related activities based on a model for attributing system activity. *Goel*, *generally*, 1-5. Forensix stores captured information in a relational database, supports queries for forensic analysis, and presents the results in human-readable form. *Goel* teaches that Forensix is implemented on Linux OS and comprises (1) an auditing module on the target system, (2) a receiver daemon and a database injector running on the backend storage system, and (3) database queries and scripts that allow replay of system activities for forensic purposes. *Goel*, 4.

1. *Goel* Identifies Activity Source, Activity Target, and Association

171. *Goel* teaches a goal of "completeness" for identifying system activity, where "*all* system-call activity is captured and can be attributed to users, processes or connections" to "accurately reconstruct *all* security incidents, immaterial of the type of attack." *Goel*, 2, 3 ("In its current implementation, all activity across the system-call interface is captured and logged."). *Goel's* target system includes an auditing module comprising a Linux kernel module that traps system calls and logs data in a kernel buffer. *Goel*, 4. The auditing module hijacks the system call table and adds stub code around several system calls to capture the system call, its timing, its parameters, its return value, and the PID of the process making the call. *Id*. The target system's auditing module traces system calls that include networking calls (e.g., *connect*, *accept*, *listen*, and *shutdown*), process management calls (e.g., *fork*, *exec*, *kill*, *exit*, and *setuid*), and file system calls (e.g., *open*, *read*, *write*, *close*). *Goel*, 4-5.

172. *Goel* teaches tracing system calls based on a model for "attributing system activity" as depicted in Figure 2, where, in one example, a process is an activity source (annotated blue) and a file (annotated red) is an activity target that the process acts on (i.e., association, annotated yellow):

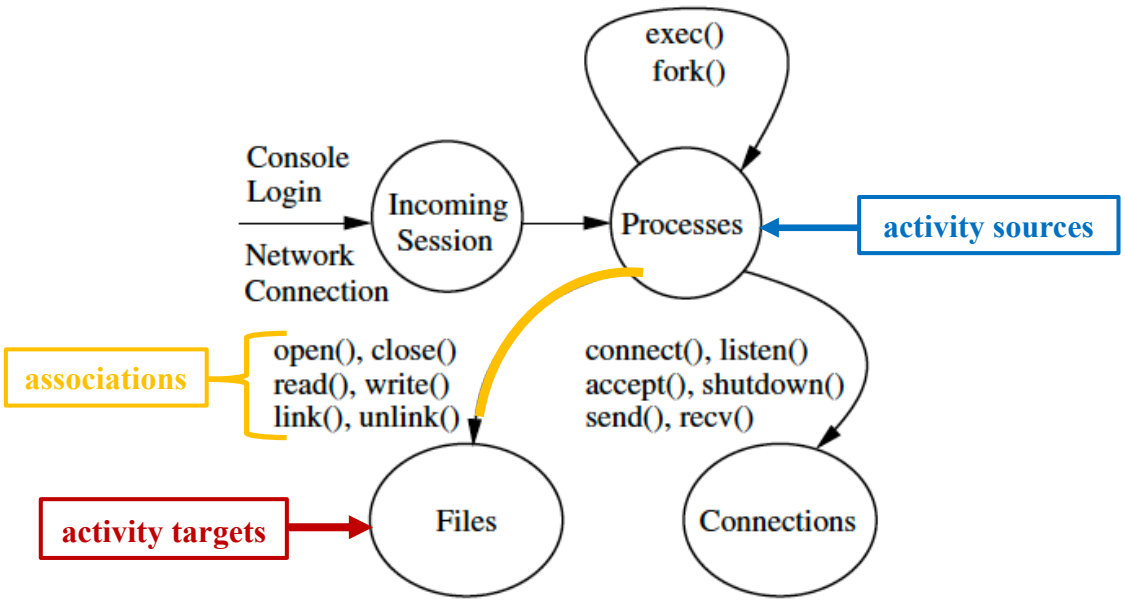


Figure 2. Relationships of system activities.

Goel, 4 ("Figure 2") (annotated), *generally*, 2-4.

173. Specifically, *Goel* teaches monitoring and capturing intrusion activity starting with a network connection or a console login and following the chain ("cascade") of related system activities including other processes, file accesses, and outgoing connections that are linked to the intrusion activity. *Goel*, 3. *Goel* teaches capturing system activities and their relationships (as depicted in figure 2) by assigning the identifier or the PID of the process that executes the activity as ownership information to each link of the graph. *Id*. At a minimum, "each system-call trace record has an associated PID and a time-stamp that helps to construct the activity relationship" and forensic queries. *Goel*, 3. Unlike previous approaches

which only use short sequences of names of system calls for intrusion detection, *Goel* teaches capturing "each system call and, among other information, its timing, parameters, return values, the process issuing the call, and the owner of that process, throughout the lifetime of the server" (i.e., target system). *Id*.

174. A POSITA would have understood that *Goel* teaches identifying a plurality of activities being performed in a computer system by attributing system activity to, for example, individual connections and sessions (as depicted in Fig. 2) and determining the events that "cascade" from an initial connection or session. *Goel*, 3. For example, as depicted in Fig. 2 above, an incoming session may execute instructions to influence one or more processes. This leads to the one or more processes (sources) performing some action (e.g., open, close, read, write) to one or more files (targets). Further, a POSITA would have understood that *Goel* teaches identifying activity sources (e.g., processes) and activity targets (e.g., files) by monitoring system calls to identify intrusions and follow as the intrusion "cascade[s] into multiple system activities including other processes, file accesses, and outgoing connections." *Goel*, 3, Fig. 2. This is because *Goel* teaches monitoring system calls in order to identify intrusions by monitoring not just individual events but how a "cascade" of events occurs over time with, for example, an incoming session affecting a process and then that process affects a file. Similarly, a POSITA would have understood *Goel* identifies actions (i.e., associations) such as read() and write()

occurring between processes and files, as depicted in Fig. 2. For example, *Goel* expressly teaches actions performed by a source (e.g., process) on a target (e.g., file). Those actions include open(), close(), read(), write(), link(), and unlink(). As discussed above, these actions meet the claimed association. *See* Section III.D.

2. Motivation to Combine *Capalik* and *Goel*

175. A POSITA would have been motivated to modify *Capalik* per *Goel's* teachings to (1) identify, and (2) store unauthorized activities including the activity source, activity target, and association of each activity. As discussed above, *Capalik* teaches a decoy system that identifies and stores a plurality of activities being performed at a VM of decoy computer 502. *Goel* teaches a forensic analysis system (Forensix) that similarly identifies and stores a plurality of activities being performed at a target computer system, wherein each of the activities includes an activity source, an activity target, and an association. *Goel*, 3-4, Fig. 2.

176. *Capalik* discloses administrators receiving "the processed information" [0011], that "paints a picture of the attack" [0012]. The generated attack signature is also presented to the admin [0013]. A POSITA reading *Capalik* would have understood that the determination that something is an attack is determined by the system, without involvement of the admin. This is because the system detects "non-standard patterns" and known abuses such as "buffer overflow attacks." A POSITA would have understood that systems are generally programmed to apply a rules

based approach that detects and categorizes certain events more efficiently than a non-automated process that relies on admin. Detecting non-standard patterns and known abuses such as buffer overflow attacks are the types of events commonly handled in an automated fashion without involving the admin.

177. However, a POSITA would have understood that these two approaches will not detect *all* abuses. Indeed, if they would, life as a security professional would be very easy: simply detect such things and block them. In reality, security professionals often see new types of abuse that the system has not identified as being bad (but are still known as bad due to indicative "side effects", e.g., due to bank accounts being accessed without authorization, suggesting the associated victim was likely either phished or had their computer infected by malware) or that are identified as having performed an unauthorized request — but it remains unknown what enabled the attacker to do so. In both of those kinds of situations, the security professional (admin) would have to analyze available data to determine what could have been the starting point of the abuse, and what particular requests should be considered indicative of the attack (these would go into the fingerprint).

178. *Capalik* is silent on how to address this problem, and does not explain what information is presented to the admin. An overabundance of information would make the job of the admin difficult. *See Goel* at Abstract. Since *Capalik* does not offer an approach for how to determine what captured information may be of

relevance to present to the admin, a POSITA would have been strongly motivated to consider the approach of *Goel*, which confirms that not all abusive behavior can be automatically detected, but that an admin needs to be able to "issue simple queries" (*Goel* at 2) to the system, to search the generated logs. This complements *Capalik*, as it would give the admin practical tools to identify new types of abuse and pinpoint particular actions of significance. This would also have improved *Capalik's* signature creation by allowing an admin to issue specific queries to generate signatures to cover new types of attacks, for example.

179. Modifying *Capalik* as proposed would have resulted in an improved IDS, as discussed above. This modification applies the well-known computer security principle that gathering more detailed threat information provides improved threat analysis. Modifying *Capalik* per *Goel's* teachings would have been use of *Goel's* known technique that would have resulted in *Capalik* identifying and storing individual attack activities (including activity sources, targets, and associations) to improve *Capalik's* similar IDS in the same manner as *Goel's* IDS.

180. *Goel* teaches that although the amount of data collected can be large, the system is feasible given the capacity of networking, CPU, and storage capacity available at the time. *Goel*, 3. *Goel* teaches that, as a result, "sacrificing some host and networking resources in order to add forensic capability will be a fairly attractive proposition." *Goel*, 3. Thus, a POSITA would have understood that even if there

were a modest increase in processing power required to modify *Capalik* per *Goel's* teachings, a POSITA would have been motivated to do so to achieve security improvements at the expense of more processing overhead required. Such a modification would have been a tradeoff that a POSITA would have been motivated to make. Further, such a modification would have resulted in an improved method and system to analyze unauthorized intrusions in a decoy system including because *Goel's* goal of "completeness" and technique for performing inquiries would have improved *Capalik's* similar system in the same way as *Goel's*. *Goel* teaches that its methods for identifying and storing an attacker's unauthorized activities (including activity sources, targets, and associations) in a computer system allow for more efficiently determining how an attacker entered the system and what the attacker did after the intrusion. *Goel*, 1. Such improvements in detection effectiveness and/or efficiency would also have improved *Capalik's* system by making *Capalik's* system more robust including for future attacks.

181. Further supporting the motivation to combine these references, as set forth in detail above, *Capalik* and *Goel* describe similar systems. For example, both teach computer systems that monitor traffic including unauthorized intrusions being performed on a computer system and identifying related activities after an intrusion. Both describe using operating systems such as Linux to interact with the unauthorized intrusion. *Capalik*, [0023]-[0024], [0040]-[0041], [0051], Fig. 5; *Goel*,

17-19. Both describe monitoring and storing information indicative of the unauthorized activities on the computer system to protect computer networks against future attacks. *See* Ground 1, Claims [1(b)]-[1(c)]; *see also* Section XI.A.1 Although *Goel* is silent on the use of a decoy system, *Goel* explains that the authors were actively working on implementing the Forensix system on a VM. *Goel*, 2. Given these similarities, a POSITA would have recognized that *Goel's* benefits would have been applicable to *Capalik*.

182. A POSITA would have had a reasonable expectation of success (REOS) when modifying *Capalik* per *Goel's* teachings for the reasons discussed below. Such a modification would have utilized *Capalik's* VMM, relational database, and visualization interface which already have monitoring, storing, and graphically displaying capability of unauthorized activities. Thus, *Capalik* already includes components for making the modifications per *Goel's* teachings. *Goel* teaches that its system implementation can be achieved by adding lines of code. *Goel*, 4-5 (explaining implementation on Linux OS, and that auditing module code is derived from the SNARE project, database is MySQL). Therefore, a POSITA would have understood that modifying *Capalik's* system (e.g., Linux-based) to additionally monitor, identify, and store each individual activity of an attack including "activity sources, activity targets, and associations" would have required straightforward modifications to program instructions/code, which would have been

well-within the skillset of a POSITA. A POSITA would have also had a reasonable expectation of success based on the similarities between *Capalik's* and *Goel's* computer devices—both in structure and in function. In particular, both *Capalik* and *Goel* teach forensic analysis systems that include monitoring and storing information about unauthorized activities performed on the computer system, and both teach utilizing Linux OS, as discussed above.

B. *Capalik-Goel* Renders Claims 2-6, 12, 15, 20, and 21 Obvious.

183. *See* Ground 1, Claims 2-6, 12, 15, 20, and 21.

C. *Capalik-Goel* Renders Claim 14 Obvious.

1. Claim 14: "The method of claim 13, wherein: instructions executed by the one or more processors have respective privilege levels; respective activity sources have respective privilege levels; and the identifying activity sources that are affected by unauthorized activities includes identifying activity sources that request to execute underprivileged instructions."

184. *See* Ground 1, Claim 13. *Capalik* teaches one or more processors. *See* Ground 1, Claim 1. As discussed below, *Capalik-Goel* one or more processors execute instructions at various privilege levels (i.e., respective privilege levels).

2. *Goel's* Teachings

185. *Goel* discloses running several local privilege escalation attacks using Forensix, and constructing a local privilege escalation query, which identifies such attacks regardless of the vulnerability they exploit by tracking a common attack pattern. *Goel*, 7-8. A common attack pattern includes using an elevated effective

user ID (eu_id) of 0 to change the attacker's real user ID (u_id) permanently to spawn a root shell. *Goel*, 8. As *Goel* explains "[l]ocal privilege escalation attacks allow a user with normal privileges to gain and retain superuser privileges illegally" and "a local root exploit takes a user process with normal privileges and converts it into a shell with superuser privileges[.]" *Goel*, 7. *Goel* teaches writing a local privilege escalation query to identify and extract root exploits that occurred in the system. *Goel*, 8. Accordingly, a POSITA would have understood that *Goel's* queries identify at least an activity source that requests to execute underprivileged instructions, as claimed.

3. Motivation to Combine *Capalik* and *Goel*

186. Modifying *Capalik* per *Goel's* teachings of using queries to identify an activity source that requests to execute underprivileged instructions would have improved *Capalik-Goel's* IDS by identifying additional unauthorized attack types, such as when activity sources request to execute underprivileged instructions. Such a modification would have improved *Capalik-Goel's* IDS by identifying when activity sources request to execute underprivileged instructions. Such a modification would have been use of *Goel's* known technique and would have improved *Capalik's* similar IDS in the same way as *Goel's* IDS. By implementing *Goel's* technique in *Capalik's* IDS, *Capalik's* attack identification and signature generation capabilities would have been improved. Thus, a POSITA would have been

motivated to utilize such techniques in *Capalik-Goel's* system. *See* Section XI.A.2 for additional motivations to combine *Capalik-Goel*, which are hereby incorporated.

187. Modifying *Capalik* to utilize *Goel's* teachings of query functionality to identify privilege escalation attempts would have required only straightforward modifications to programming instructions/code for the same reasons discussed above. As discussed above, a POSITA would have been motivated to modify *Capalik* per *Goel's* teachings and these modifications would have had a reasonable expectation of success for the reasons discussed above. *See* Section XI.A.2.

XII. GROUND 3: *CAPALIK* ALONE OR IN VIEW OF *GOEL* AND IN FURTHER VIEW OF *KING* RENDERS CLAIMS 7-11, 14, 16, 18-19 OBVIOUS

A. Claim 7: "The method of claim 1, further comprising: graphically displaying at least a subset of the stored activities."

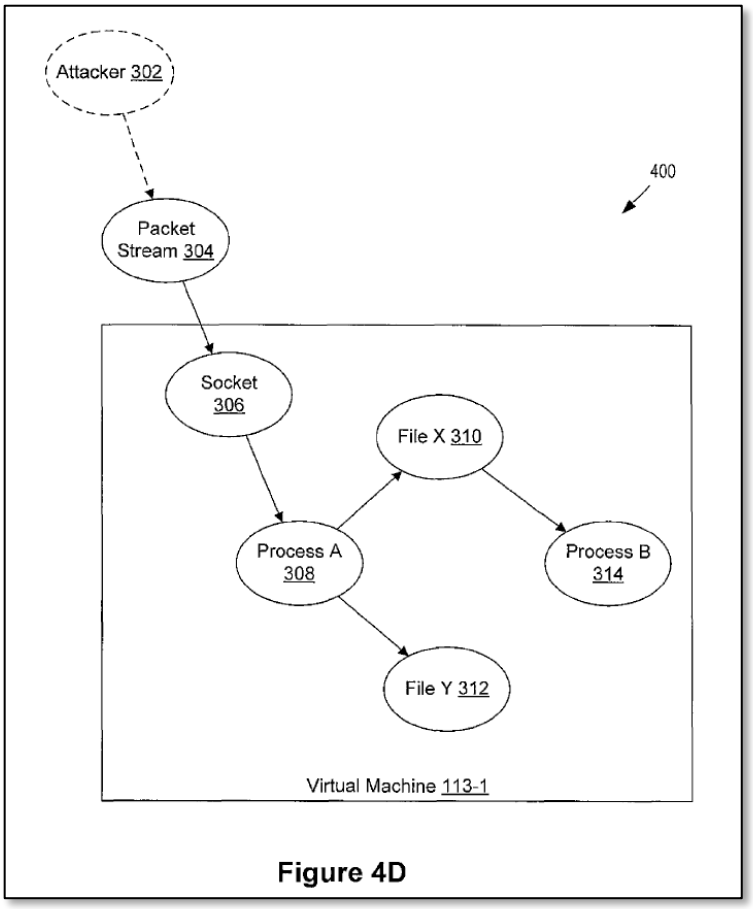
188. As explained above with respect to Claim [1(b)]-[1(c)], *Capalik* (Ground 1) or *Capalik-Goel* (Ground 2) teaches identifying a plurality of activities (including activity sources, activity targets, and associations) performed at the computer system, and retaining detailed forensic data that can be used to generate signatures indicative of the activities. As explained below, *King* teaches graphically displaying the stored activities, including a subset of the stored activities.

1. *'872 Patent's* graphically displaying a subset of the stored activities

189. The '872 Patent describes "the decoy network device graphically displays (626) at least a subset of the stored activities. For example, the graphical

representation can be an activity diagram as shown in FIGS. 3A-3D and 4A-4E."

'872 Patent, 16:62-65.



'872 Patent, Fig. 4D.

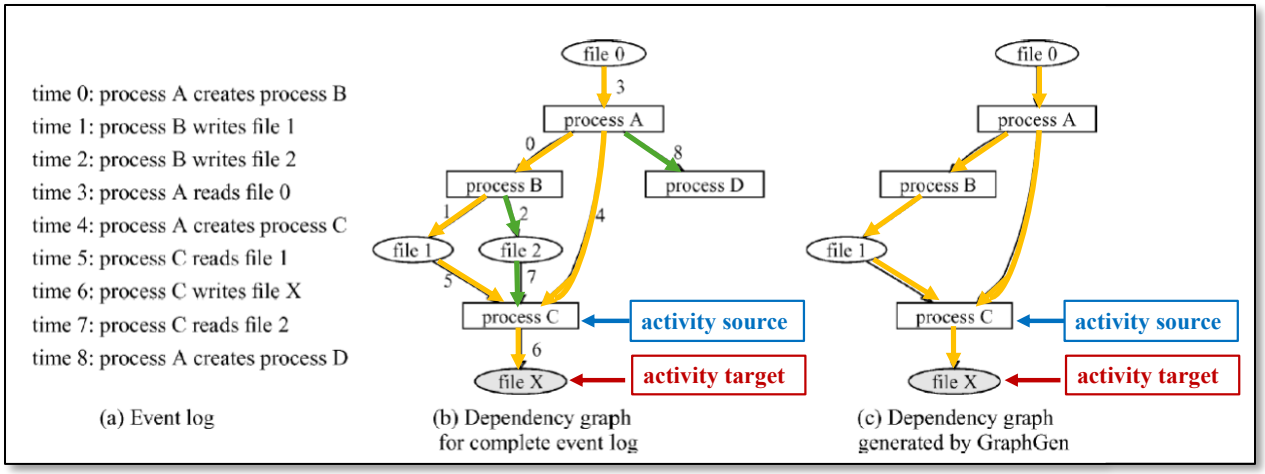
2. *King* teaches graphically displaying a subset of stored activities

190. *King* teaches analyzing unauthorized activities using operating-system-level (OS-level) information-flow graphs to highlight an attacker's activities inside a VM by following the flow of information from one object to another, and the dependency-causing events between objects. *King*, 10, 13. *King* discloses that its "EventLogger" identifies and stores information-flow events in real-time, and that

its "GraphGen" processes EventLogger logs to identify and graphically display information-flow graphs of an intrusion. *King*, 12. *King* discloses dependency graphs that "represent the flow of information either backward in time or forward in time[,]" depending on which of *King's* systems (e.g., Backtracker or ForwardTracker) was used to generate the graph. *King*, 17.

191. *King's* ForwardTracker functions like the BackTracker except that it provides a forward causal analysis. *King*, 21 ("Backward information-flow graphs add events to a graph if they affect an object before that object's time threshold, forward tracking adds events to a graph if they are affected by an object after that object's time threshold (Figure 3.1)."). Accordingly, a POSITA would have understood that *King's* ForwardTracker is analogous to the '872 Patent's network diagram illustrated in Figs. 3A-3D, which also represent the flow of information forward in time.

192. In sum, a POSITA would have understood that whether utilizing *King's* Backtracker or ForwardTracker systems, the EventLogger and GraphGen components would have identified, stored, and graphically displayed a plurality of activities being performed at the virtual machine, wherein each of the activities includes an activity source, target and association.



King, Fig. 3.2 (annotated).

193. As illustrated, GraphGen generates an information-flow graph (i.e., dependency graph) by detecting a suspicious object (e.g., file X having the wrong contents at time 10, not depicted in Fig. 3.2) and following the attack back in time from this suspicious "detection point" (time 10), tracking the sequence of events and objects that led to the suspicious object. *King*, 17-18 (noting the "detection point" for this Backtracker flow is time 10–the time at which "the administrator knows that file X has the wrong contents"). *King's* Figure 3.2(b) depicts the log of events as a complete dependency graph including all the events that GraphGen considers relevant to the affected file X. *King*, 17-18. As annotated above, process C represents one activity source that acts on file X the activity target. Figure 3.2(c) depicts a filtered graph, ignoring events and objects that do not affect file X by time 6. *King*, 18-19. For example, as discussed below, Fig. 3.2(c) displays only malicious actions that could have an effect on Process C's action on file X.

194. A POSITA would have understood that *King's* GraphGen graphically displays a plurality of activities performed at the VM wherein each of the activities includes an activity source, activity target and an association between the activity source and activity target. For example, *King's* GraphGen "build[s] an information-flow graph that depicts the dependency relationships between all objects seen over that execution." *King*, 16. These graphs contain the source (e.g., "process C"), the target (e.g., "file X"), and the respective association (e.g., "writes[,]" "reads[,]" "creates"). *King*, Fig. 3.2. A POSITA would have understood each event depicted from time 0 to time 8 in Fig. 3.2(a) describes actions (e.g., reading/writing) between respective activity sources and targets. *'872 Patent*, 13:34-38.

195. Namely, as I explain below, *King's* method to create backward dependency graphs determines the relevance of each logged event (i.e. action) based on time intervals and whether the event affects the detection point (i.e. file X having wrong contents at time 10). *King*, 17-20, Fig. 3.2. As depicted in Fig. 3.2, *King's* GraphGen considerers all related events that occurred before the detection point at time 10, and prunes away those events that do not affect the detection point. *King*, 17-19. Further, a POSITA would have understood that *King's* Fig. 3.2(b) identifies and displays associations between activity sources and activity targets of each activity, wherein the associations (i.e. actions) between activity sources and targets that affect file X by time 10 are malignant (annotated yellow), and the associations

that do not affect file X by time 10 are benign (annotated green). *King's* system prunes away benign associations, resulting in a filtered graph (Fig. 3.2(c)) depicting only activities with malignant associations.

196. *King's* Fig. 3.2 illustrates dependency relationships between objects, wherein one object affects the state of another object by performing an action (e.g., process C affecting file X by writing to file X). *King*, Fig. 3.2. Each arrow in Figure 3.2(b) depicts the action itself between the objects, and the dependency relationship between the objects. *King* teaches tracking the actions between objects by time interval to identify the dependency relationship. *King*, 13. This strategy decreases false dependencies by clearly establishing causality between the objects. *Id.*

197. *King's* GraphGen logs and maps actions or events between objects in a time-dependent manner by assigning a time threshold to the object associated with the detection point. *King*, 17-18. Backward tracking adds events to a graph if they affect an object before that object's time threshold, and forward tracking adds events to a graph if they are affected by an object after the object's time threshold. *King*, 21. Accordingly, those events and objects that do not affect the detection point at time 10 may be pruned away to create the filtered dependency graph Fig. 3.2(c).

198. *King's* filtered graph is then displayed to a system administrator, allowing the administrator to more efficiently analyze intrusions in a computer system. *King*, 26-28. For example, the administrator is seeing only those sources,

targets, and actions that are associated in a time-dependent manner with the malicious action. Without filtering using *King's* backtrack system, the administrator would instead see significantly more unaffiliated sources, targets, and actions while attempting to decipher whether the signature should be saved in the IDS.

199. Figs. 3A-3D and 4A-4D of the '872 Patent show mapping unauthorized activities between objects. *'872 Patent*, Figs. 3A-3D, 4A-4D, 11:23-12:26. *King's* Fig. 3.2 similarly shows mapping of unauthorized activities between objects. *King*, Fig. 3.2, 16-20. Each mapping between objects includes labeling the activity source and activity target. Accordingly, a POSITA would have understood *King's* dependency graphs depicted in Fig. 3.2 are similar to Figs. 3A-3D, 4A-4D of the '872 Patent and graphically display at least a subset of the stored activities, including activity sources, targets, and associations.

3. Motivation to Combine *Capalik-King/Capalik-Goel-King*

200. For the reasons explained below, a POSITA would have been motivated to modify *Capalik/Capalik-Goel* per *King* to display a subset of the stored activities, including activity sources, activity targets, and associations for each activity. As discussed above, *Capalik/Capalik-Goel* teaches a decoy computer that identifies, and stores activities performed at the decoy computer. *See* Claim [1(b)]-[1(c)]. As explained in detail below, *King* teaches a similar decoy computer that identifies, stores, and graphically displays attack activities including the activity sources,

activity targets, associations, and an indication of which individual activities are malignant/malicious. *See King*, 10-13, 17-18, Fig. 3.2.

201. As discussed above, *Capalik* teaches a decoy system that identifies and stores a plurality of activities being performed at a VM of a decoy computer system. *King* teaches a similar decoy system that identifies and graphically displays a plurality of activities being performed at the VM and provides a more detailed disclosure about each identified activity including an activity source, an activity target, and an association. *See King*, Fig. 3.2. Additionally, *King* teaches storing the activity source and target in memory. *King*, Fig. 3.2, 13 (describing logging of "a source object, a sink object, and a time interval" with EventLogger).

202. Modifying *Capalik/Capalik-Goel* as proposed would have resulted in an improved IDS. For example, in the case of *Capalik* and *King*, the generated signature is reviewed by an administrator who decides to advance the signature to the IDS/IPS. *Capalik*, [0059]; *King*, 72. Modifying *Capalik/Capalik-Goel* to graphically identify a subset of activities (e.g., malicious activities) would improve accuracy of the signatures flagged by the administrator, decreasing false negatives and false positives. False negatives lead to the administrator or system incorrectly discarding (or not including) attacks rather than storing attack information at the IDS (or including the information in a signature), leaving the system vulnerable to similar future attacks. Alternatively, false positives would bog down the system, requiring

any future activity to be compared against multiple signatures that do not actually match malicious activity. False positives may also cause the blocking of benevolent activity, which is a common concern among enterprise users of security systems, and a common reason for slow enterprise roll-outs of security measures. Decreasing incorrectly labeled signatures would enhance the IDS analysis of future activity that matches activity of the maintained signatures.

203. This modification applies the well-known computer security principle that gathering more detailed threat information provides improved threat analysis. As stated above, *Capalik*/*Capalik-Goel* would have stood to gain from *King's* method of distinguishing malicious actions from benign actions. Modifying *Capalik/Capalik-Goel* in this way would provide a more detailed map of the threat and improve signature generation and prevention of subsequent attacks.

204. Furthermore, displaying a subset of actions would have allowed an administrator to more readily identify the attack path. *King* explains that by identifying "the events and objects that could have affected that detection point" BackTracker helps administrators "find and focus on a few important objects and events to understand the intrusion." *King*, 72. By separating out the pertinent subset of sources, targets, and actions, the program significantly decreases the load on the administrator to trace the attack. *Capalik's* system may generate an attack signature for review by an administrator to decide whether to send the signature to the IDS or

IPS. *Capalik*, [0059]. A POSITA would have been motivated to incorporate this advantage into *Capalik/Capalik-Goel* to aid the administrator in their review of the signature for the same reasons expressed by *King*. Such modifications would decrease the load on the administrator, including the time required for review. This would result in faster deployments of updated signatures, which in turn results in improved security and lower losses.

205. *King* teaches its system has "low overhead" for processing power needed to run and perform analysis of its decoy system. *King*, 8-11. Thus, a POSITA would have understood that even if an increase in processing power were required to modify *Capalik/Capalik-Goel* per *King's* teachings, such an increase would have been worth it to achieve security improvements. Accordingly, a POSITA would have been motivated to do so to achieve security improvements at the expense of more processing overhead required. Specifically, a POSITA would have recognized that small increases in processing requirements at *Capalik/Capalik-Goel's* decoy computer 502 would not have significantly impacted the overall performance of the network. Additionally, a POSITA would have recognized that the advantages of threat prevention to *Capalik/Capalik-Goel's* network far outweigh any potential, although unlikely, modifications necessary to manage increased processing requirements.

206. Modifying *Capalik/Capalik-Goel* per *King* would have been use of a known technique of identifying and graphically displaying activity source, target, and association to improve *Capalik's/Capalik-Goel's* similar system in the same way as *King's* system. Namely, as shown below, *Capalik/Capalik-Goel* teaches a VM and VMM system for identifying sources, targets, and each respective association. *Capalik/Capalik-Goel* further teaches providing an attack signature to an administrator. *Capalik*, [0059]. As such, modifying *Capalik/Capalik-Goel* to display *King's* chart would be modifying a similar system in the same way as *King*. Detailed above, such a modification would have resulted in an improved analysis of unauthorized intrusions in a decoy system. *King* teaches its methods for identifying, storing, and graphically displaying an attacker's unauthorized activities (including activity sources, targets, and associations) in a decoy system allow more efficiently determining which application contained the vulnerability that the attacker exploited; further, *King* teaches preventing future intrusions/attacks by updating or disabling any vulnerable applications. *King*, 2. *King's* system is able to improve identification of vulnerable applications by specifically removing benign actions and only displaying objects which were susceptible to the malicious actions. *King*, 2, 73. As explained above, the modified *Capalik/Capalik-Goel* system would also have appreciated these benefits.

207. Further supporting the motivation to combine these references, as set forth in detail above, *Capalik/Capalik-Goel* and *King* teach similar systems. All describe monitoring unauthorized activities being performed on VMs, set up as decoy computer systems, to protect computer networks against zero-day attacks. *See*, Claim [1(b)]-[1(c)]; *King*, 22-25. All teach using virtualized OS(s) such as Linux to interact with the unauthorized intrusion. *Capalik*, [0040]-[0041]; *King*, 22-23. All describe storing information indicative of the unauthorized activities on the VM in a database within the decoy system. *See*, Claim [1(b)]-[1(c)]; *King*, 13, Fig. 3.2. *Capalik* and *King* describe displaying information indicative of the unauthorized activities on the VM using visualization interfaces. *Capalik* [0059]; *King*, Fig. 3.2. Given these similarities, a POSITA would have recognized modifying *Capalik/Capalik-Goel* per *King* would have achieved *King's* security benefits. For these same reasons, a POSITA would have had a reasonable expectation of success when modifying *Capalik/Capalik-Goel* per *King's* teachings.

208. Such a modification would have utilized *Capalik's/Capalik-Goel's* VMM, relational database, and visualization interface which already have monitoring, storing, and graphically displaying capabilities for unauthorized activities. *King* also teaches GraphGen "runs on any computer used by the administrator[.]" *King*, 25. Similarly, GraphGen "does not require any data from the computer in question other than the EventLogger log" which was simply

implemented by adding only 40 lines of code and required "no changes to the guest operating system." *King*, 23-25. Therefore, a POSITA would have understood that modifying *Capalik's/Capalik-Goel's* system to additionally graphically display the stored activities (including activity sources, activity targets, and associations), as taught by *King*, would have required only straightforward modifications to program/code instructions, which would have been well-within the skillset of a POSITA. Namely, *Capalik/Capalik-Goel* and *King* both run on Linux-based operating systems. *Capalik*, [0040]-[0041]; *King*, 22-23. Both operating systems are similarly configured to operate on virtual machines. *Capalik*, [0040], Fig. 5; *King*, 22-23, Fig. 3.3. Accordingly, a POSITA would have recognized that the *Capalik/Capalik-Goel* system would have been readily modified with *King's* teachings to work in the same way.

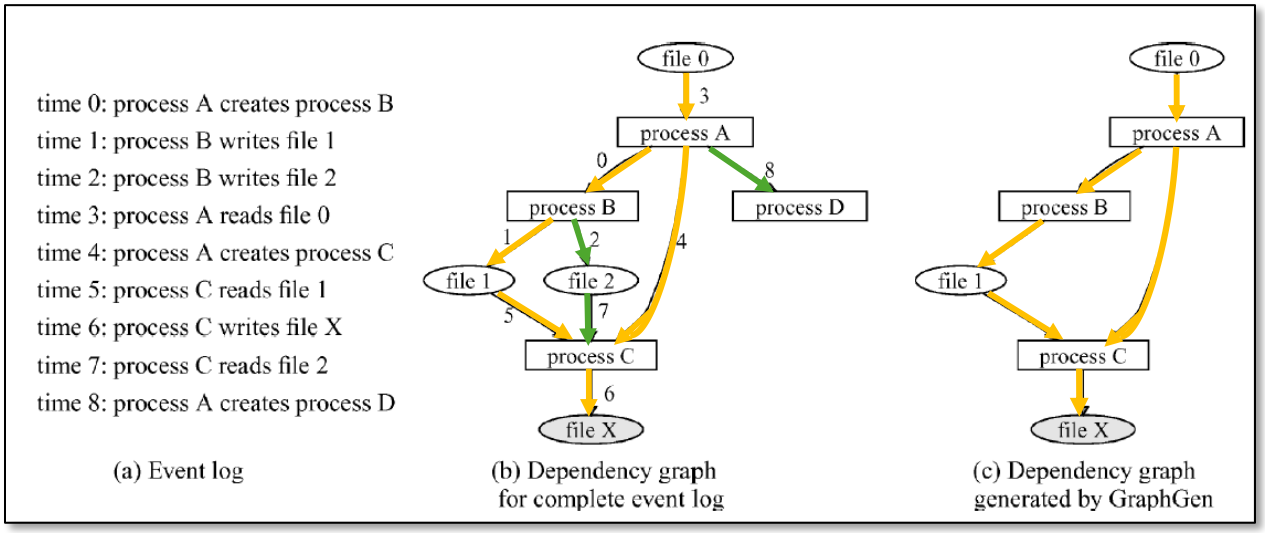
B. Claim 8: "The method of claim 7, wherein the displaying includes visually distinguishing at least a subset of the associations."

209. As explained above in Claim 7, *Capalik-King/Capalik-Goel-King* teaches dependency graphs that graphically display a subset of the stored activities. As explained below, *King's* dependency graphs further visually distinguish at least a subset of the associations.

210. The '872 Patent explains that when the type of associations is identified (e.g., malignant, benign), at least a subset of the associations can be visually distinguished (e.g., a bold arrow representing a malignant association; a regular

arrow representing a benign association), and that any graphical scheme can be used to visually distinguish the subset of the associations. *'872 Patent*, 12:32-41.

211. As explained for Claim 7, *King's* complete dependency graph (Fig. 3.2(b)) depicts malignant associations (annotated yellow) and benign associations (annotated green). And the filtered graph (Fig. 3.2(c)) depicts a subset of the associations, including only malignant associations (annotated yellow):



King, Fig. 3.2 (annotated). Accordingly, a POSITA would have understood that *King's* dependency graphs visually distinguish a subset of the associations by depicting only malignant associations in a filtered dependency graph. As explained in Claim 7, this subset comprises the malicious actions that could have led to process C's action on file X based on each actions time interval. *See* Claim 7. Accordingly, a POSITA would have understood that *Capalik-King/Capalik-Goel-King* teaches

graphically displaying at least a subset of the stored activities, wherein the displaying includes visually distinguishing at least a subset of the associations.

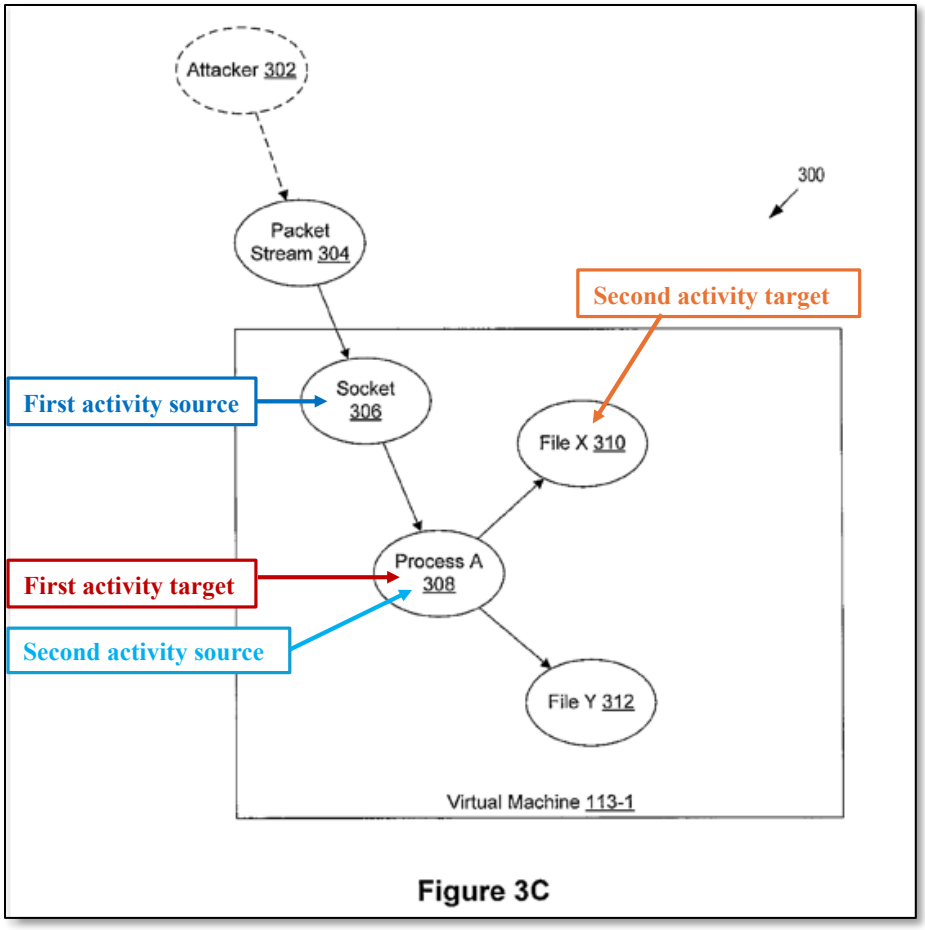
212. As discussed above, a POSITA would have been motivated to modify *Capalik/Capalik-Goel* per *King's* teachings and would have had a reasonable expectation of success. *See* Claim 7.

C. Claim 9: "The method of claim 1, wherein an activity target of a first activity of the plurality of activities is an activity source of a second activity of the plurality of activities."

213. As explained above, *King*'s dependency graphs identify a subset of activities of the plurality of activities wherein each activity includes an activity source, activity target, and association. *See* Claim 7. As explained below, *Capalik-King/Capalik-Goel-King* further teaches that the subset of activities includes a first activity and a second activity, and the activity target of the first activity is the activity source of the second activity.

1. '872 Patent's Teachings

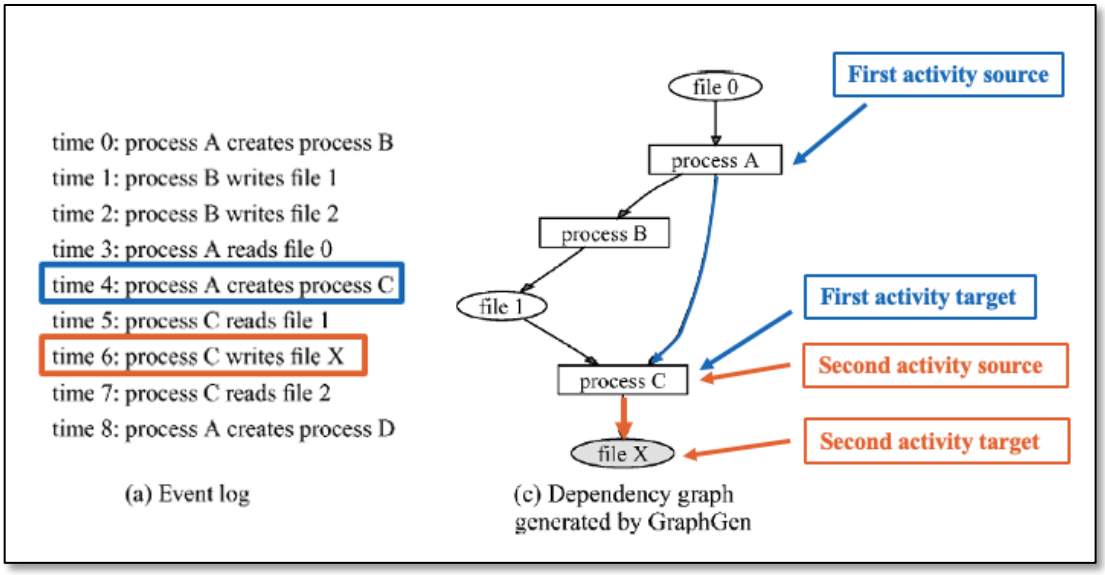
214. The '872 Patent points to Figure 3C (below) as an exemplary embodiment of this Claim:



'872 Patent, 17:8-19, Fig. 3C (annotated). The '872 Patent explains figure 3C illustrates a first activity between socket 306 and process A 308 (i.e., data from socket 306 is directed to process A 308) and a second activity between process A 308 and file X 310 (i.e., process A 308 accesses file X). Process A 308 is the activity target of the first activity (annotated red) and the activity source of the second activity (annotated light blue). *Id.*, 17:11-15.

2. *King's* Teachings

215. Similar to the '872 Patent, *King* depicts a similar subset of activities in dependency graph 3.2(c):



King, Fig. 3.2, (annotated). The annotations above highlight a first unauthorized association (blue) between a first activity source (process A) and a first activity target (process C). Namely, "process A creates process C" at time 4. The annotations above also highlight a second unauthorized association (orange) between a second activity source (process C) and a second activity target (file X). Namely, "process C write file X" at time 6. Accordingly, *King* teaches a first activity and a second activity where ***process C*** is the activity target of the first activity and the activity source of the second activity.

3. Motivation to Combine *Capalik/Capalik-Goel* per *King*

216. As discussed above, a POSITA would have been motivated to modify *Capalik/Capalik-Goel* per *King's* teachings and would have had a REOS. Those arguments are incorporated here as well. *See* Claim 7. Additionally, a POSITA would have been motivated to modify *Capalik/Capalik-Goel* per *King* to track

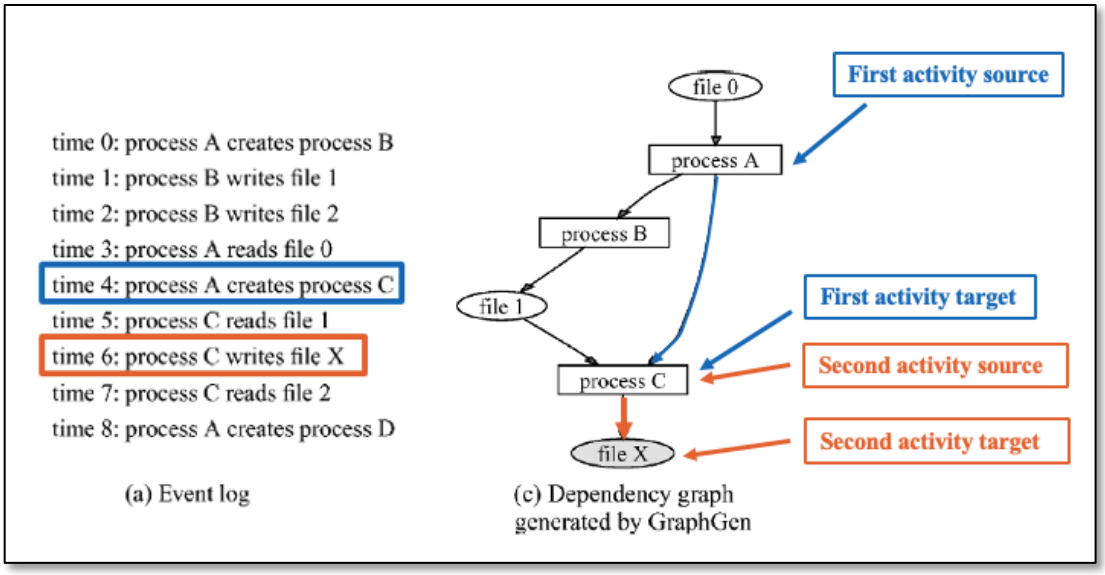
dependencies over time to improve attack detection and achieve improved attack signatures. Such a use of a known technique of tracking dependencies/activities over time would have improved *Capalik/Capalik-Goel* in the same way as *King* including by identifying attack patterns and improving attack signatures to help prevent future attacks.

D. Claim 10: "The method of claim 1, further comprising: identifying a respective association between a first activity source and a first activity target as unauthorized, wherein the first activity target, when associated as a second activity source with a second activity target, causes unauthorized activities on the second activity target."

217. As explained above, *Capalik*/*Capalik-Goel* teaches identifying a plurality of activities performed at a computer system, wherein each activity includes an activity source, activity target, and association. *See* Claim [1(b)]. As explained below, *Capalik-King/Capalik-Goel-King* further teaches (i) identifying a respective association between a first activity source and a first activity target as unauthorized, and (ii) that unauthorized activity is caused between the second activity source and target when the first activity target is the second activity source.

1. *King's* Teachings

218. *King* depicts a subset of activities and associations in dependency graph 3.2:



King, Fig. 3.2 (annotated). The annotations above highlight a first unauthorized association (blue) between a first activity source (process A) and a first activity target (process C). Namely, "process A creates process C" at time 4. The annotations above also highlight a second unauthorized association (orange) between a second activity source (process C) and a second activity target (file X). Namely, "process C write file X" at time 6. As set forth above regarding Claim 8, all associations depicted in *King's* Fig. 3.2(c) are unauthorized/malignant. Thus, a POSITA would have understood that both associations annotated above are unauthorized.

219. Accordingly, *Capalik-King/Capalik-Goel-King* teaches (i) identifying a respective association between a first activity source (e.g., process A) and a first activity target (e.g., process C) as unauthorized, and (ii) that unauthorized activity is caused between the second activity source (e.g., process C) and target (e.g., file X).

In at least the example discussed above, process C is both the first activity target and the second activity source, as claimed. *See also* Claims 7-9.

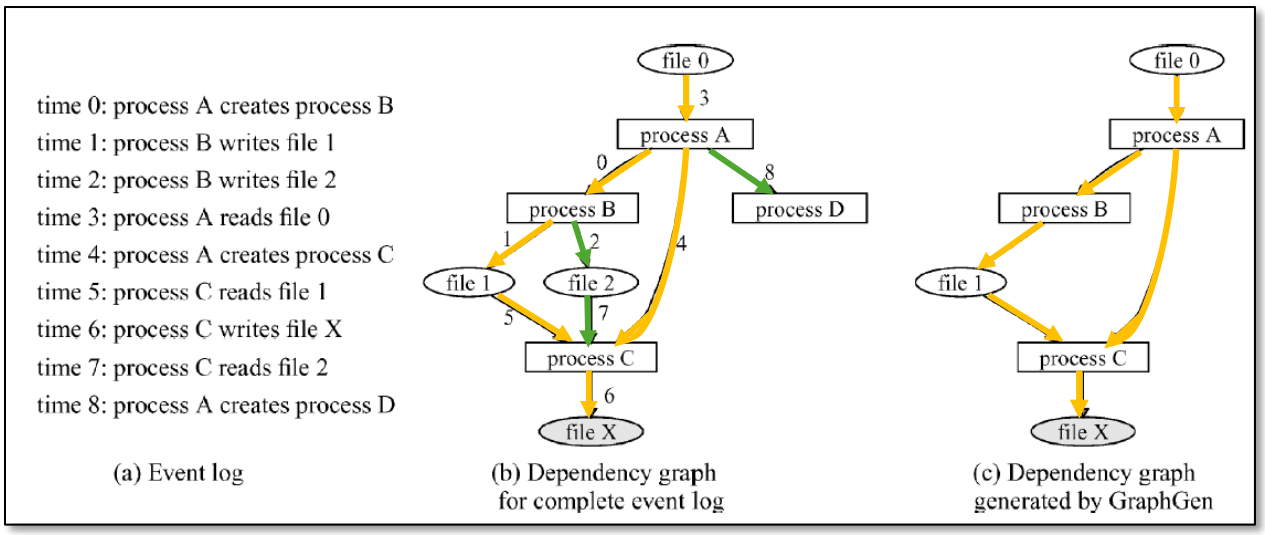
2. Motivation to Combine *Capalik/Capalik-Goel* per *King*

220. As discussed above, a POSITA would have been motivated to modify *Capalik/Capalik-Goel* per *King's* teachings and would have had a REOS. Those arguments are incorporated here as well. *See* Claims 7, 9.

221. A POSITA would have been motivated to modify *Capalik/Capalik-Goel* per *King* to track dependencies over time to improve attack detection and achieve improved attack signatures. Such a use of a known technique of tracking dependencies/activities over time would have improved *Capalik/Capalik-Goel* in the same way as *King*. Such a modification would have improved *Capalik/Capalik-Goel's* IDS in the same way as *King's* IDS by identifying a respective association between a first activity target and first activity source as unauthorized when the first activity target is associated as a second activity source with a second activity target causes unauthorized activities on the second activity target. Indeed, such a modification would have utilized *King's* teachings of dependency tracking over time to determine such associations between sources and targets (wherein a first target may become a second source for some attacks). Such a modification would have had a reasonable expectation of success for the reasons discussed above. *See* Claims 7-9.

E. Claim 11: "The method of claim 10, further comprising: tracking a transfer of malicious associations over time."

222. *See* Claim 10. As explained above, *Capalik-King/Capalik-Goel-King* teaches identifying and tracking a plurality of activities over time, wherein each activity includes an activity source, activity target, and an association. *See* Claims 7-10. As explained in Claims 7-10, *King's* dependency graphs distinguish between malignant and benign associations, and a filtered graph (Fig. 3.2(c)) depicts only malignant associations:



King, Fig. 3.2 (annotated); *see* Claims 7-10. For the same reasons stated above at Claims 7-10, a POSITA would have been motivated to modify *Capalik/Capalik-Goel* per *King* to track a transfer of malicious associations over time.

223. Additionally, a POSITA would have been motivated to modify *Capalik/Capalik-Goel* per *King* to track a transfer of malicious associations over time to improve attack detection and achieve improved attack signatures. Such a use

of a known technique of tracking a transfer of malicious associations over time would have improved *Capalik/Capalik-Goel* in the same way as *King*. Namely, *King's* IDS tracks the transfer of malicious associations over time, improving attack detection and ensuring that future attacks are prevented. This additional temporal viewpoint of attacks provides useful information for detecting attacks and preventing against similar attacks, and such a viewpoint would be beneficial to the *Capalik/Capalik-Goel* systems just as it is to *King*. Indeed, such a modification would have utilized *King's* teachings of dependency tracking to track malicious associations over time to determine such malicious associations between sources and targets (wherein a first target may become a second source for some attacks, for example). Such a modification would have had a reasonable expectation of success for the reasons discussed above. *See* Claims 7-10.

F. Claim 14: "The method of claim 12, wherein: instructions executed by the one or more processors have respective privilege levels, respective activity sources have respective privilege levels; and the identifying activity sources that are affected by unauthorized activities includes identifying activity sources that request to execute underprivileged instructions."

224. *See* Claim 12. *Capalik/Capalik-Goel* teaches one or more processors. *See* Claim 1. To the extent it is argued *Capalik-Goel* does not teach Claim 14, *Capalik-King/Capalik-Goel-King* teaches Claim 14.

1. *King's* teachings

225. As discussed below, *Capalik-King's/Capalik-Goel-King's* one or more processors execute instructions at respective privilege levels. *King* discloses a simulated attack where Backtracker generates a dependency graph that identifies activity sources requesting to execute underprivileged instructions. *King*, 35, Fig. 4.5. The attack depicted in Figure 4.5 gains unprivileged access via httpd, then downloads and compiles a program (sxp) that "takes advantage of a local exploit against sendmail" to overflow an argument buffer and provide a root shell. *King*, 35. The unprivileged httpd and sxp request to execute underprivileged instructions because an unprivileged user should not be able to provide a root shell or create a privileged user account, but the exploit here allows an unprivileged user to do exactly that: provide a root shell and create a privileged user that then executes privileged instructions. *King*, 35. The root shell is used to add a privileged user to the password files, log into the machine using this new user, and modify /etc/xinetd.conf (the detection point for this attack). *Id*.

226. A POSITA would have understood that *King's* graph in Fig. 4.5 identifies at least an activity source that requests to execute underprivileged instructions, as claimed. Namely, a POSITA would have understood that the httpd activity source running on *King's* system is unprivileged, as indicated by the attack "gain[ing] unprivileged access via httpd[.]" *King*, 35. Additionally, the sxp program

that is downloaded and compiled is an unprivileged activity source that uses an argument buffer overflow to provide a root shell. The root shell creates a privileged user, which is a higher privilege level than either the sxp or httpd activity sources. The attack utilizes the created privileged user to modify the xinetd.conf file, which is the detection point for this attack. A POSITA would have understood that modifying /etc/xinetd.conf would have had a respective privilege level different from that of the httpd or sxp activity sources as this modification required privileged user access. *Id.* Thus, a POSITA would have recognized that modifying /etc/xinetd.conf represents a request to execute underprivileged instructions that were only available to a privileged user. Since each of these activity sources, as well as others, were identified by *King's* BackTracker system, a POSITA would have understood *Capalik-Goel-King* teaches Claim 14.

2. Motivation to Combine *Capalik/Capalik-Goel* and *King*

227. A POSITA would have been motivated to modify *Capalik-Goel* to utilize *King's* technique of identifying activity sources that are affected by unauthorized activities including identifying activity sources that request to execute underprivileged instructions. Such a modification would have improved *Capalik/Capalik-Goel's* IDS by identifying when activity sources request to execute underprivileged instructions. Such a modification would have been use of *King's* known technique and would have improved *Capalik/Capalik-Goel's* similar IDS in

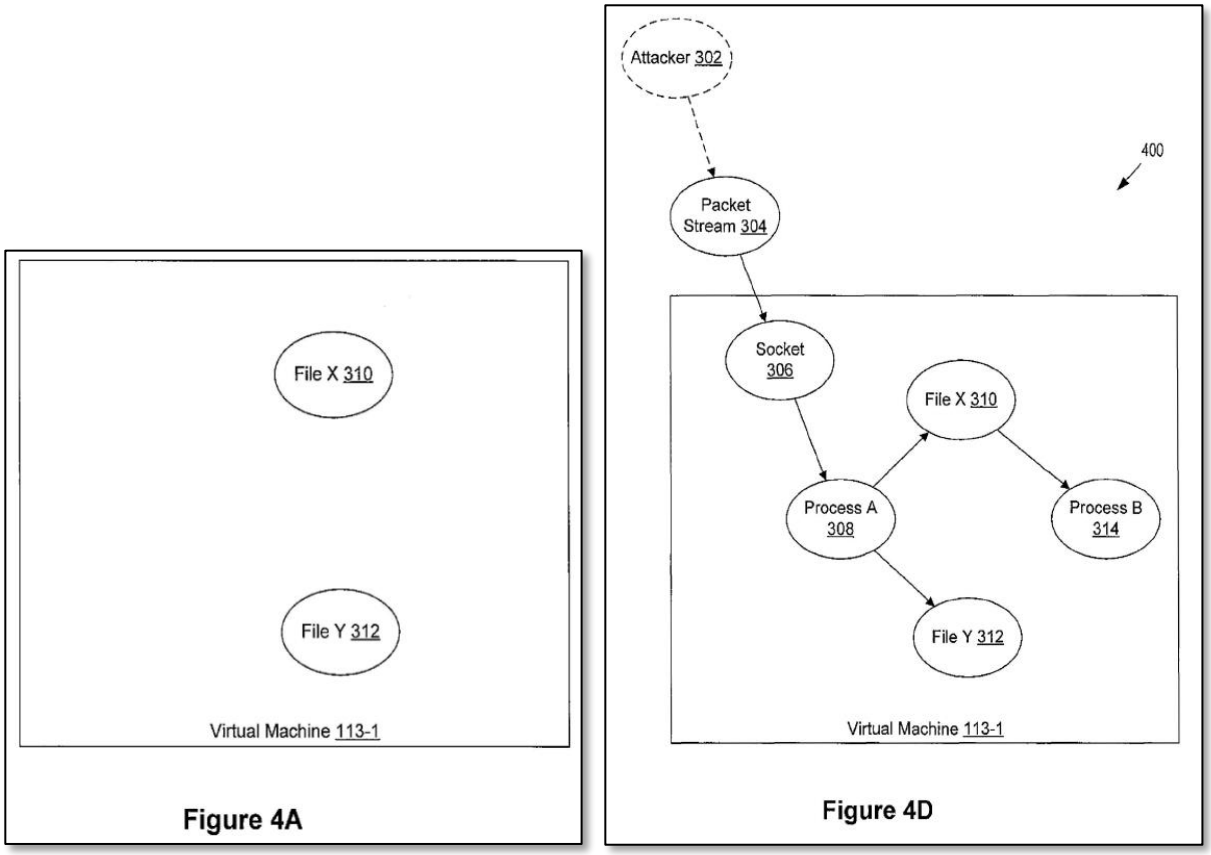
the same way as *King's* IDS. By implementing *King's* technique in *Capalik/Capalik-Goel's* IDS, *Capalik's/Capalik-Goel's* attack identification and signature generation capabilities would have been improved. Thus, a POSITA would have been motivated to utilize such techniques in *Capalik-King's/Capalik-Goel-King's* system. *See* Claims 7-11 for additional motivations to combine *Capalik-King/Capalik-Goel-King*, which are hereby incorporated.

228. Modifying *Capalik/Capalik-Goel* to utilize *King's* teachings of query functionality to identify privilege escalation attempts would have required only straightforward modifications to programming instructions/code for the same reasons discussed above. As discussed above, a POSITA would have been motivated to modify *Capalik/Capalik-Goel* per *King's* teachings, and these modifications would have had a reasonable expectation of success for the reasons discussed above. *See* Claims 7-11.

G. Claim 16: "The method of claim 1, further comprising: determining an activity status for the stored events, the activity status comprising one of: a complete status, representing that a source of unauthorized activities is identified, and each activity target in the stored activities is associated with the source through one or more associations; and an incomplete status, which represents that the source of unauthorized activities is not identified, or at least one activity target in the stored activities is not associated with the source through one or more associations."

1. '872 Patent's Description of Complete Status and Incomplete Status

229. The ’872 Patent describes an example of a complete status is depicted in Fig. 4D, wherein “each activity is associated with the source, packet stream 304 through one or more associations[.]” *’872 Patent*, 17:23-28. In contrast, Fig. 4A illustrates an incomplete status in which at least one target is not associated with a source. These incomplete (left) and complete (right) statuses are depicted below:

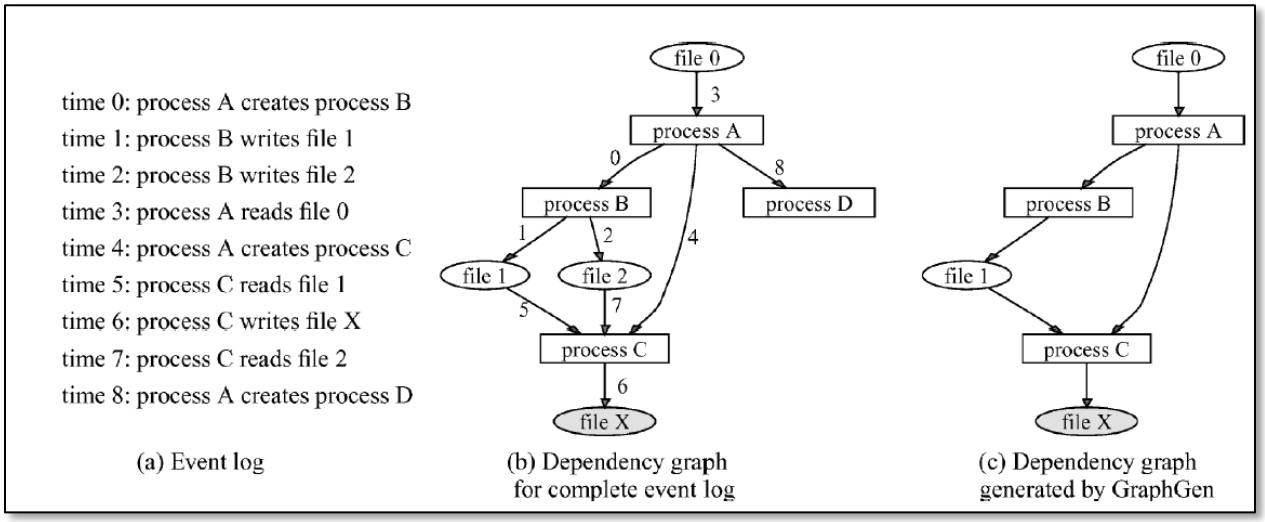


’872 Patent, Figs. 4A, 4D.

2. *King’s* Teachings

230. As explained above with respect to Claim [1(c)]-[1(e)], *Capalik*/*Capalik-Goel* teaches identifying, storing, and displaying unauthorized activities wherein each of the activities includes an activity source, target, and

association. Like Fig. 4D in the '872 Patent, *King's* Fig. 3.2(c) illustrates a complete activity status for an attack. As illustrated below, *King's* system identifies a source of unauthorized activities (e.g., file 0) and that source's relation to every activity target (e.g., process B, file 1, process C, file X) through one or more identified associations.



King, Fig. 3.2.

231. Namely, *King* teaches that for Backward dependency, "GraphGen reads the logs of events" and for "each event, GraphGen evaluates whether that event can affect any object that is currently in the dependency graph." *King*, 17. As such, every association between each source and each target is identified so GraphGen can then remove associations occurring at a time interval after the time interval of interest. This process would have been understood as a complete status, as claimed, since all associations between each source and target would be identified as malicious or

benign before finalizing the graph indicating the path of the invasion. Accordingly, *Capalik-King*/*Capalik-Goel-King* teaches determining a complete activity status for the stored events.

3. Motivation to Combine *Capalik/Capalik-Goel* per *King*

232. A POSITA would have been motivated to modify *Capalik/Capalik-Goel* per *King*. *See* the motivations to combine and reasons for reasonable expectation of success when making such a combination discussed above. *See* Claim 8. Such a modification would have been use of *King's* known technique to determine a complete activity status for stored events and would have improved *Capalik's/Capalik-Goel's* similar IDS in the same way as *King's*.

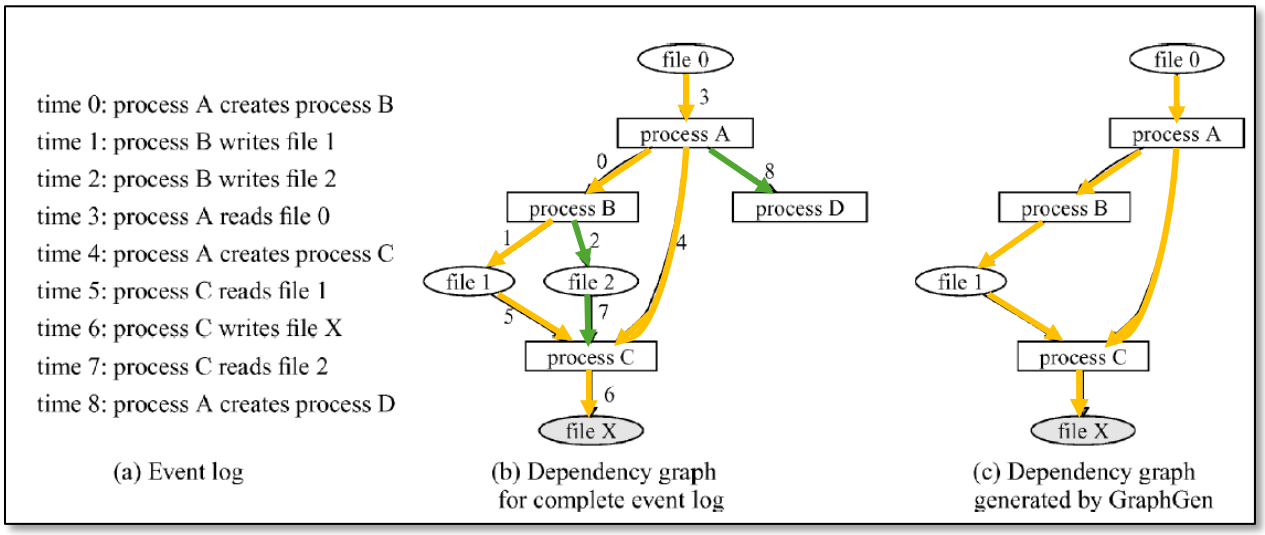
233. Modifying *Capalik/Capalik-Goel* would have been a straightforward modification requiring only changes to programming instructions. Therefore, a POSITA would have had a reasonable expectation of success when making such a modification. *See* the reasons for reasonable expectation of success at Claims 7-10, 14 above.

H. Claim 18: "The method of claim 1, further comprising: identifying a respective activity as unauthorized based on a determination that the respective activity is associated with an unauthorized activity that is distinct from the respective activity."

234. *See* Claims 9-10.

I. Claim 19: "The method of claim 1, wherein the association includes a type that represents whether or not the activity source has modified the activity target to perform the unauthorized activities."

235. As explained above in Claim 7, *Capalik-King/Capalik-Goel-King* teaches identifying a type of the association between the activity source and the activity target, as depicted in *King's* dependency graph generated by GraphGen:



King, Fig. 3.2 (annotated); *see also* Claim 7. As discussed above in Claim 7, *King's* GraphGen generates an information-flow graph (i.e., dependency graph) by detecting a suspicious object (e.g., file X having the wrong contents at time 10) and following the attack back in time from this suspicious "detection point" (time 10), tracking the sequence of events and objects that led to the suspicious object. *King*, 17-18. *King's* Figure 3.2(b) depicts the log of events as a complete dependency graph including all activities occurring before the suspicious content was identified in file X. *King*, 17-18. Figure 3.2(c) depicts a filtered graph, ignoring events and objects

that do not affect file X as of time 10. *King*, 18-19. In other words, *King's* filtered dependency graph in Fig. 3.2(c) removes any associations that did not contribute (directly or indirectly) to the unauthorized activity identified at time 10 (i.e., suspicious content written into file X). A POSITA would have understood that *King's* Fig. 3.2(b) identifies and displays associations between activity sources and activity targets of each activity, wherein the associations that affect file X by time 10 are malignant (annotated yellow), and the associations that do not affect file X by time 10 are benign (annotated green). *See* Claims 7, 16. As explained in Claim 7, *King* teaches pruning away activities that include benign associations, resulting in a filtered graph (Fig. 3.2(c)) depicting only activities that include malignant associations and are thus unauthorized. A POSITA would have understood that the associations remaining after this filtering are of a type that represents a source modifying its target to perform unauthorized activities either directly (e.g., writing incorrect content into file X) or indirectly (e.g., causing a subsequent source to write incorrect content into file X). *See* Claim 7 for motivations to combine *Capalik/Capalik-Goel* and *King* as well as reasons for a reasonable expectation of success when making such a combination.

XIII. GROUND 4: *CAPALIK* ALONE OR IN VIEW OF *GOEL*, AND IN FURTHER VIEW OF *PIKE* RENDERS CLAIM 13 OBVIOUS

A. Claim 13: "The method of claim 12, wherein the identifying activity sources that are affected by unauthorized activities includes identifying activity sources that request to access a portion of memory that is set as non-executable."

236. As explained above, *Capalik* (Ground 1) or *Capalik-Goel* (Ground 2) teaches Claims 1, 12. *Capalik* expressly teaches monitoring for memory-related errors such as buffer overflow attacks. *Capalik*, [0057]. But *Capalik* does not discuss using NX (i.e., no execute) bit technology or explain how buffer overflow attack sources are identified. As set forth below, a POSITA would have been motivated to supplement *Capalik* (or *Capalik-Goel*) with *Pike's* NX bit technology techniques, which are particularly effective against buffer overflow attacks.

1. *Pike's* teachings

237. *Pike* teaches that NX bit technology techniques are particularly effective against buffer overflows attacks from internet worms. *Pike*, 3:12-21. As explained below, *Capalik-Pike/Capalik-Goel-Pike* teaches identifying activity sources affected by unauthorized activities that request to access a portion of memory that is set as non-executable.

238. The '872 Patent describes identifying activity sources that request to access a portion of memory that is set as non-executable can be achieved by data execution prevention (DEP) mechanisms, including using a non-executable bit. *'872 Patent*, 16:8-23 (describing DEP mechanisms were well-known). For example, the

’872 Patent describes “microprocessors manufactured by Intel Corporation (Santa Clara, Calif.) allow setting one or more portions of the memory as non-executable (e.g., using a non-executable bit in respective page table entries), and generate an exception when there is an attempt to execute an instruction that is set as non-executable.” *’872 Patent*, 16:8-20. In another example, “operating systems sold by Microsoft Corporation (Redmond, Wash.) has an analogous function implemented in the operating system (e.g., Windows XP, Windows 7, etc.).” *’872 Patent*, 16:20-23. Thus, a POSITA would have understood that DEP mechanisms were known in the prior art.

239. As discussed below, *Pike* teaches one such DEP mechanism. *Pike* discloses scanning “input strings” (e.g., data packets), identifying subsequences in the input strings that could be maliciously exploited, and configuring the target computer system to trigger a fault (or other triggered event) when certain execution conditions suggest use of data or code implicated by the identified subsequences. *Pike*, 3:12-19, 5:13-38, 8:65-67, 2:7-33. *Pike* discloses several techniques to trigger an action when memory of the target system is accessed by the potentially dangerous data in the input string. *Pike*, 8:65-9:5.

240. *Pike* teaches setting a relevant no-execute bit that triggers an exception. *Pike*, 8:65-9:12, 9:24-10:1, 12:58-13:22. An exception handler is used to evaluate the likelihood that a particular exception was caused by a dangerous interpretation

of a received input string. *Id*. A POSITA would have understood that potentially dangerous input strings (e.g., data packets) are activity sources affected by unauthorized activities. As discussed above, data streams, files, and other data packets/input strings are activity sources. *See* Claim [1(b)]. *Pike's* input strings also would have been potentially malicious data packets or other input strings. Further, a POSITA would have understood that *Pike's* technique of setting a relevant no-execute bit to identify potentially dangerous input strings is a DEP mechanism that identifies activity sources affected by unauthorized activities. In particular, *Pike's* no-execute bit technique is one of the examples of DEP described in the '872 Patent. *'872 Patent*, 16:8-20. Similarly, *Pike's* technique would have identified sources affected by unauthorized activities. Accordingly, a POSITA would have understood that *Pike* teaches identifying activity sources affected by unauthorized activities that request to access a portion of memory that is set as non-executable, as claimed.

2. Motivation to Combine *Capalik-Pike/Capalik-Goel-Pike*

241. A POSITA would have been motivated and found it obvious to modify *Capalik/Capalik-Goel* per *Pike's* teachings such that *Capalik-Pike/Capalik-Goel-Pike* would have identified activity sources affected by unauthorized activities that request to access a portion of memory that is set as non-executable. As discussed above, *Pike* teaches such a DEP technique of using a non-executable bit (NX bit) that would have identified activity sources that try to access non-executable portions

of memory including by triggering an exception. *Pike*, 8:65-9:12, 9:24-10:1, 12:58-13:22. As explained in Claim 12, *Capalik/Capalik-Goel* teaches identifying a plurality of unauthorized activity sources but is silent regarding identifying unauthorized activity sources that request access to a portion of memory of the target computer system. A POSITA would have recognized that modifying *Capalik/Capalik-Goel* per *Pike's* teachings would have allowed identifying a larger variety of unauthorized activity sources, including those that employ infection mechanisms that request access to the memory of a target system. Such a modification would have been use of a known technique that would have provided *Capalik/Capalik-Goel* with NX bit (DEP) technology to further enhance *Capalik/Capalik-Goel's* similar security system and protect against more attacks in the same way as *Pike*.

242. Additionally, *Capalik* addresses events that would be indicative of abuse, e.g., buffer overflow attacks, but does not clarify how exactly a buffer overflow attack would be detected. A POSITA wishing to implement *Capalik* would look for suggestions how to do this. *Pike* discloses such suggestions, as it discloses methods using NX bit technology to detect attempts to execute data, which is what is typically at the heart of buffer overflow attacks.

243. Specifically, *Pike* discloses using a variety of techniques to identify the activity sources that request access to a system's memory, one of these techniques

includes using no-execute bit (also known as NX bit) technology for malware or attack detection, which entails marking certain portions of the system's memory as non-executable. *Pike*, 8:65-9:5. As discussed above, NX bit technology is one form of DEP. Thus, such modification would have been use of a known technique (DEP) to identify unauthorized activity sources that request to access a portion of memory that is set as non-executable to improve *Capalik's/Capalik-Goel*'s similar system in the same way as *Pike.* Although supplying an NX bit could impose overhead on the system, a POSITA would have been motivated to accept more overhead in exchange for more robust malware attack detection. Such a modification would have been a tradeoff that a POSITA would have been motivated to make to improve the robustness of *Capalik/Capalik-Goel's* IDS/IPS.

244. A POSITA would have had a REOS modifying *Capalik/Capalik-Goel* per *Pike's* teachings because *Capalik/Capalik-Goel* already teaches similar hardware topology (e.g., one or more processors, memory, VM) to support *Pike's* functionality of identifying activity sources that access portions of memory set as non-executable. *See* Claim [1(Pre)]; *Pike*, 6:19-24. Modifying *Capalik/Capalik-Goel* per *Pike* to apply a known technique (NX bit/DEP) would have required straightforward modifications to software/programming instructions, which would have been well within the skillset of a POSITA.

XIV. GROUND 5: *CAPALIK* ALONE OR IN VIEW OF *GOEL*, AND IN FURTHER VIEW OF *FARLEY* RENDERS CLAIM 17 OBVIOUS

A. Claim 17: "The method of claim 1, further comprising: determining an activity level representing a frequency of unauthorized activities on the first computer system during a predefined time interval."

245. As explained above, *Capalik* (Ground 1) or *Capalik-Goel* (Ground 2) teaches Claim 1. In particular, *Capalik/Capalik-Goel* teaches monitoring attacker activity on the computer system of Decoy Computer 502. *See* Claim [1(a)]. Additionally discussed above, *Capalik/Capalik-Goel* teaches identifying a plurality of unauthorized activities including an activity source, activity target, and an association between the activity source and activity target. *See* Claim [1(b)]. Further, *Capalik/Capalik-Goel* generates a signature/fingerprint that is indicative of activity on the VM. *See* Claims [1(d)], 3. As discussed above, when generating a fingerprint/signature, *Capalik/Capalik-Goel* teaches either performing the process manually (e.g., an admin decides to generate a signature) or automatically (where the system automatically generates a signature). *See* Claim [1(d)].

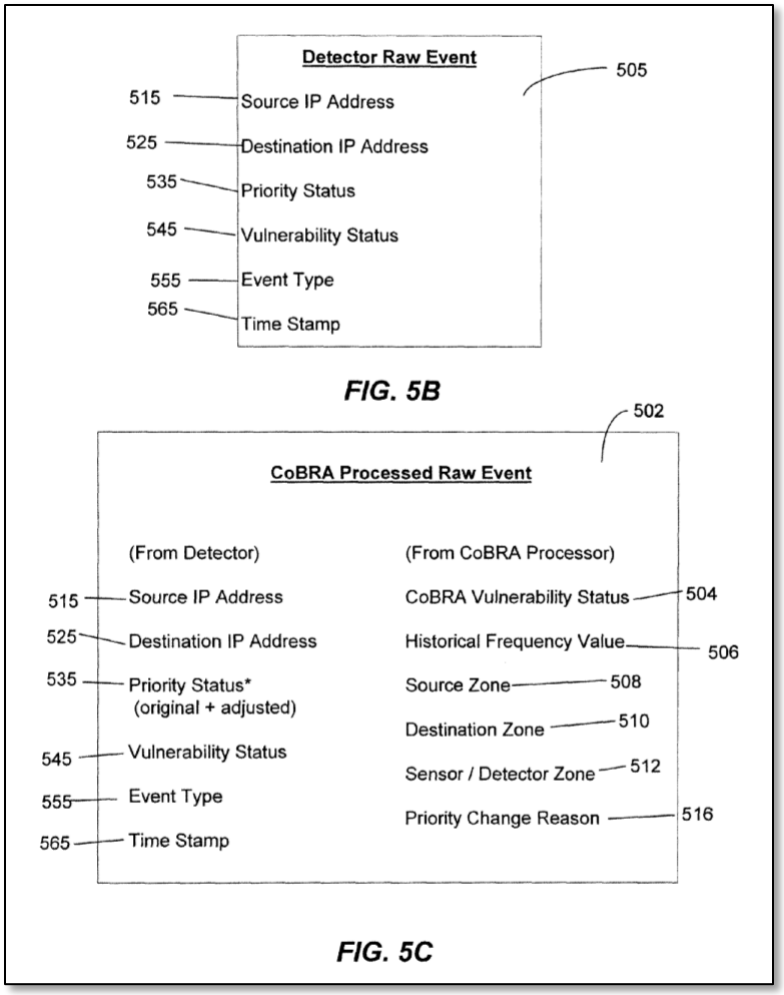
246. In the combination below with *Farley*, *Capalik/Capalik-Goel* is modified to determine a historical frequency value ("activity level") for the unauthorized traffic/activity at the VM of Decoy Computer 502. Also discussed below, such a modification would have improved *Capalik/Capalik-Goel's* IDS, including when deciding whether to generate a signature, by also allowing either the

system and/or admin to rely on historical frequency value information in the analysis.

1. *Farley's* teachings

247. As explained below, *Farley* teaches determining an activity level (i.e. historical frequency value) representing a frequency of unauthorized activities on a computer system during a predefined time interval.

248. *Farley* teaches assembling suspicious computer activity ("raw events") from multiple data sources (e.g., IDS/IPS, firewalls, audit systems). *Farley*, 3:25-4:11. *Farley* teaches analyzing assembled activity to detect relationships between raw events that may indicate malicious behavior ("mature correlation event"), and provides an organized data presentation, as depicted below:



Farley, Figs 5B-5C, 3:25-46, 4:18-34. Figure 5B illustrates exemplary raw event 505 that an IDS detector generates. *Farley*, 11:65-67. Raw event 505 may comprise source IP address 515, destination IP address 525, priority status 535, detector assigned vulnerability status 545, event type parameter 555, and time stamp 565. *Farley*, 11:67-12:4.

249. *Farley* also teaches that the priority status of a raw event may be adjusted by a CoBRA (Context Based Risk Adjustment) processor by comparing the raw event to information stored in the context database such as historical event

frequency values. *Farley*, 15:11-36, Fig. 5C. For example, Fig. 5C includes Historical Frequency Value 506. *Farley*, 12:13-27, 24:36-39 ("Historical frequency data is used by the fusion engine to aid in distinguishing events caused by normal non-malicious network activity from those caused by unusual and malicious activity."), Fig. 5C. A historical frequency value "can be a number of events per unit time, such as events per day, or a mathematically related value, such as an average time between events … [and] typically indicates how frequently raw events of a particular type from a particular source machine to a particular destination machine are seen on the network being monitored[.]" *Farley*, 24:27-39.

250. *Farley* teaches using a historical frequency value to indicate a frequency of raw events of a particular type that have been monitored on the network, and a priority status can be assigned based on the historical frequency value. *Farley*, 27:34-28:18. The historical frequency can be compared to a frequency event threshold value, and a determination can be made that a raw event is likely to be a non-malicious event, for example, if the historical frequency is higher than a threshold. *Id.* On the other hand, using the historical frequency value compared with a threshold can determine an event is likely to be a malicious event if the historical frequency is lower than a threshold, for example. *Id.* Accordingly, a POSITA would have understood that *Farley* teaches measuring the frequency of events, categorized by source, target, action, etc., occurring at the system. A POSITA would have

understood that *Farley* then uses the frequency value to help determine whether a specific event is malicious or benign.

2. Motivation to Combine *Capalik-Farley/Capalik-Goel-Farley*

251. A POSITA would have recognized the importance of computational efficiency for the *Capalik* system, especially given the need to surveil all traffic to and on a given system. Recognizing that there are many potential entry points for abusive behavior (e.g., through an open port, as part of an email, in web traffic, in the context of user-installed software, etc.), a POSITA would have appreciated the need to process large amounts of data from multiple "sensors" in an efficient manner, or the monitoring functionality of *Capalik* may become a computational bottleneck of the system, rendering the protection less effective. Furthermore, a POSITA would have recognized that the computational burden of monitoring large numbers of internal potential targets would increase the burden of this system, along with the processing of all the data being observed and reported. *Farley* addresses the problem of efficient processing of multiple independent streams of data and would have been important to consider in any system with the architecture described in *Capalik*, especially during an era in which low-cost cloud computing was not an available option. This motivation to combine would be independent of whether the combination is with *Capalik* or *Capalik-Goel*, as neither of these expressly addresses the computational cost.

252. A POSITA would have been motivated and found it obvious to modify *Capalik/Capalik-Goel* per *Farley* such that *Capalik-Farley/Capalik-Goel-Farley* would have determined an activity level (i.e., historical frequency value) representing a frequency of unauthorized activities on the VM. *Farley*, 24:27-39, 27:34-28:18. Modifying *Capalik/Capalik-Goel* as proposed would have utilized a known technique to track historical frequency values of intrusions/attacks and would have improved *Capalik/Capalik-Goel's* IDS in the same way as *Farley's* by tracking unauthorized activity levels on the VM. Such tracking would have improved the *Capalik/Capalik-Goel* IDS including for the reasons discussed above, such as efficiently processing large amounts of data.

253. Further, *Farley* teaches a historical frequency value (and/or other environmental information) can be used to adjust the priority status of an identified suspicious unauthorized activity, providing a more efficient risk assessment of suspicious activity in a computer system. *Farley*, 15:11-36, 23:53-60. Utilizing the historical frequency value in this way would have allowed for more efficient processing of the data over time. For example, *Capalik/Capalik-Goel's* system would have benefitted from prioritizing which attacks are placed in front of the admin to determine which to send in an attack signature. *Capalik*, [0056], [0059]. As taught by *Farley*, *Capalik/Capalik-Goel* would prioritize displaying actions with a lower historical frequency value as these were more likely to be malicious. *Farley*,

27:34-28:18. The ’872 Patent describes a similar benefit for determining an activity level representing a frequency of unauthorized activities. *’872 Patent*, 10:20-26.

254. As discussed above, *Capalik/Capalik-Goel* teach a decision (whether performed manually by an admin or automatically by the system) to generate a signature/fingerprint. *See* Claim [1(d)]. Modifying *Capalik/Capalik-Goel* per *Farley’s* teachings would also have improved *Capalik/Capalik-Goel’s* decision-making process for whether to generate a signature by including an analysis including one or more historical frequency values. That is, a POSITA would have been motivated to take the historical frequency value into consideration when generating a signature because the historical frequency value would have indicated how often a particular event occurred and would have helped track patterns of events.

255. Further, the proposed modification adheres to recognized computer security principles—i.e., gathering more detailed threat information provides improved threat analysis and detection where limited data acquisition can hinder performance and result in unperceived threats. In particular, tracking events over time helps provide a better understanding of what activities an attacker is engaging in at the VM. Although determining a historical frequency value could result in increased overhead in *Capalik-Farley/Capalik-Goel-Farley*, a POSITA would have been motivated to trade increased overhead for these IDS security improvements.

As discussed above, *Farley's* technique would have provided more efficient processing for large amounts of data. However, even if an increase in processing overhead were to occur, a POSITA would have been motivated to achieve better security as a tradeoff of increasing processing overhead. Thus, the proposed modification would have used a known technique of determining a historical frequency value of suspicious/unauthorized activities to improve IDS forensic analysis and would have benefited *Capalik/Capalik-Goel's* similar system in the same way as *Farley's*.

256. A POSITA would have had a REOS modifying *Capalik/Capalik-Goel* per *Farley*. *Capalik/Capalik-Goel* already teaches computer hardware (e.g., processor, memory), modules (e.g., processing module) and networks for performing *Farley's* technique. *Capalik/Capalik-Goel* already teaches using an IDS/IPS as well. Such a modification would have utilized *Capalik's/Capalik-Goel's* processing module that generates an attack report that already includes attack-identifying information from the VM. Therefore, a POSITA would have understood that modifying *Capalik's/Capalik-Goel's* system to additionally capture "a historical frequency value" would require only straightforward modifications to programming instructions, which was well within the skillset of a POSITA. A POSITA would have also had a REOS based on the functional similarities between *Capalik's/Capalik-Goel's* IDS and *Farley's* IDS. For example, *Capalik/Capalik-Goel* and *Farley* teach

intrusion detection systems that monitor and store information about suspicious/unauthorized activities).

XV. CONCLUSION

257. I hereby declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct, and that all statements made of my own knowledge are true and that all statements made on information and belief are believed to be true. I understand that willful false statements are punishable by fine or imprisonment or both. *See* 18 U.S.C. § 1001. If called to testify as to the truth of the matters stated herein, I could and would testify competently.

Respectfully submitted:

By:_____
Dr. Markus Jakobsson

Date: October 31, 2024.

CV and Research Statement

Markus Jakobsson
www.linkedin.com/in/markusjakobsson
www.markus-jakobsson.com

1 At a Glance

- **Focus.** *Identification of security problems, trends and solution along four axes* – computational*,* structural*,* physical *and* social*; quantitative and qualitative fraud analysis; development of disruptive security technologies.*
- **Education.** *PhD* (Computer Science/Cryptography, University of California at San Diego, 1997); *MSc* (Computer Science, University of California at San Diego, 1994); *MSc* (Computer Engineering, Lund Institute of Technology, Sweden, 1993).
- **Large research labs.** *San Diego Supercomputer Center* (Researcher, 1996-1997); *Bell Labs* (Member of Technical Staff, 1997-2001); *RSA Labs* (Principal Research Scientist, 2001-2004); *Xerox PARC* (Principal Scientist, 2008-2010); PayPal (Principal Scientist of Consumer Security, Director, 2010-2013); Qualcomm (Senior Director, 2013-2015); Agari (Chief Scientist, 2016–2018); Amber Solutions Inc (Chief of Security and Data Analytics, 2018 – 2019); ByteDance (Principal Scientist, 2020-2021)
- **Academia.** *New York University* (Adjunct Associate Professor, 2002-2004); *Indiana University* (Associate Professor & Associate Director, 2004-2008; Adjunct Associate Professor, 2008-2016).
- **Entrepreneurial activity.** *ZapFraud* (Anti-scam technology; CTO and founder, 2012-current); *RavenWhite Security* (Authentication solutions; CTO and founder, 2005-); *RightQuestion* (Consulting; Founder, 2007-current); *FatSkunk* (Malware detection; CTO and founder, 2009-2013 – FatSkunk was acquired by Qualcomm); *LifeLock* (Id theft protection; Member of fraud advisory board, 2009-2013); *CellFony* (Mobile security; Member of technical advisory board, 2009-2013); *PopGiro* (User Reputation; Member of technical advisory board, 2012-2013); *MobiSocial* (Social networking, Member of technical advisory board, 2013); *Cequence Security* (Anti-fraud, Member of technical advisory board, 2013–current)
- **Anti-fraud consulting.** *KommuneData* [Danish govt. entity] (1996); *J.P. Morgan Chase* (2006-2007); *PayPal* (2007-2011); *Boku* (2009-2010); *Western Union* (2009-2010).

APPENDIX A
IPR2025-00070
CrowdStrike EX1003 Page 179

- **Intellectual Property, Testifying Expert Witness.** *Inventor of 100+ patents; expert witness in 25+ patent litigation cases* (McDermott, Will & Emery; Bereskin & Parr; WilmerHale; Hunton & Williams; Quinn Emanuel Urquhart & Sullivan; Freed & Weiss; Berry & Domer; Fish & Richardson; DLA Piper; Cipher Law Group; Keker & Van Nest). Details and references upon request.

- **Publications**. Books: *Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft* (Wiley, 2006); *Crimeware: Understanding New Attacks and Defenses* (Symantec Press, 2008); *Towards Trustworthy Elections: New Directions in Electronic Voting* (Springer Verlag, 2010); *Mobile Authentication: Problems and Solutions*)Springer Verlag, 2012); *The Death of the Internet* (Wiley, 2012); *Understanding Social Engineering* (Springer Verlag, 2016); *Security, Privacy and User Interaction* (Springer Verlag, 2020); *100+ peer-reviewed publications*

2 At a Glance

Ten years before Bitcoin was created, I formalized the notion of Proof of Work and described its use for mining of crypto payments. I later developed energy-efficient alternatives to this paradigm, and showed how to enable mining on mobile devices, which is not possible for Bitcoin. I am the founder of the academic discipline of phishing and have developed techniques to predict fraud trends years before they emerge, enabling countermeasures to be developed before they are needed. I developed the notion of implicit authentication, which is now ubiquitous; I also founded a company that developed the first retroactive virus detection technology, with guarantees of detection; the company was acquired by Qualcomm in 2013. I have worked as chief scientist and similar positions in startups as well as industry behemoths, such as PayPal. I have several hundred patents to my name and am a prominent security researcher with hundreds of peer reviewed publications and an array of textbooks. My 1997 PhD thesis, from University of California at San Diego, was on distributed electronic payment systems with revocable privacy.

3 Work History (Highlights)

1. **Member of Technical Staff, Bell Labs (1997-2001).** Markus was part of the security research group at Bell Labs. He formalized the notion of *proof of work*, later an integral part of BitCoin.

2. **Principal Scientist, RSA Labs (2001-2005).** Markus posited that phishing would become a mainstream problem, and developed ethical techniques for identifying likely trends based on human subject experiments.

3. **Associate Professor, Indiana University (2005-2008).** Markus was hired to lead the newly formed security group at Indiana University, and

created a research group comprising approximately 10 professors and 30 students, studying social engineering and fraud.

4. **Principal Scientist, Xerox PARC (2008-2010).** Markus was hired to lead the research efforts of the Xerox PARC security group, and developed the notion of *implicit authentication*, a technology that is now ubiquitous.

5. **Principal Scientist, PayPal (2010-2013).** Markus did research on security and user interfaces, and developed techniques to reduce the losses associated with *liar buyer fraud*.

6. **Senior Director, Qualcomm (2013-2016).** Qualcomm acquires FatSkunk, a company founded by Markus. At FatSkunk, Markus developed *retroactive* malware detection with provable security guarantees. A simplified version of this is now deployed with almost all Qualcomm chipsets.

7. **Chief Scientist, Agari (2016-2018).** Markus developed a technique to acquire fraudster intelligence by compromising scammer email accounts – *while staying within the law* – resulting in the extradition of several African scam lords to the U.S.

8. **Chief of Security and Data Analytics, Amber Solutions (2018-2020).** Markus developed usable configuration methods supporting improved security and privacy for IoT installations.

9. **Chief Scientist, ByteDance (2020-2021).** Markus oversaw the establishment of a research group and a research agenda at ByteDance, and contributed to their intellectual property and product security.

10. **Chief Scientist, Artema LABS (2021-).** Managing the research, product strategy, and patent strategy for Artema LABS, a Los Angeles based startup in the Crypto/NFT sector.

4 Publication List

Books (1-8); book chapters, journals, conference publications and other scientific publications (9-147), issued /published U.S. patents (148-234). For an updated list, and for international patents, please see `www.markus-jakobsson.com/publications` and appropriate patent search engines.

References

[1] M. Jakobsson, *Security, Privacy and User Interaction*, ISBN 978-3-030-43753-4, 110 pages, 2020.

[2] M. Jakobsson, *Understanding Social Engineering Based Scams*, ISBN 978-1-4939-6457-4, 130 pages, Springer, 2016.

[3] M. Jakobsson, *Mobile Authentication: Problems and Solutions*, ISBN 1461448778, 125 pages, Springer, 2013.

[4] M. Jakobsson, (editor) *The Death of the Internet*, ASIN B009CN2JVE, 359 pages, IEEE Computer Society Press, 2012.

[5] D. Chaum, M. Jakobsson, R. L. Rivest, P. Y. Ryan, J. Benaloh, and M. Kutylowski, (editors), *Towards Trustworthy Elections: New Directions in Electronic Voting*, 411 pages, (Vol. 6000), Springer, 2010.

[6] M. Jakobsson and Z. Ramzan (editors), *Crimeware: Trends in Attacks and Countermeasures*, ISBN 0321501950, Hardcover, 582 pages, Symantec Press / Addison Wesley, 2008.

[7] M. Jakobsson and S. A. Myers (editors), *Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft*, ISBN 0-471-78245-9, Hardcover, 739 pages, Wiley, 2006.

[8] M. Jakobsson, M. Yung, J. Zhou, *Applied Cryptography and Network Security: Second International Conference , Yellow Mountain, China, 2004*, 511 pages, Lecture Notes in Computer Science (Book 3089), 2004.

[9] M. Jakobsson, "Permissions and Privacy," in IEEE Security & Privacy, vol. 18, no. 2, pp. 46-55, March-April 2020

[10] M. Jakobsson, "The Rising Threat of Launchpad Attacks," in IEEE Security & Privacy, vol. 17, no. 5, pp. 68-72, Sept.-Oct. 2019

[11] J Koven, C Felix, H Siadati, M Jakobsson, E Bertini, "Lessons learned developing a visual analytics solution for investigative analysis of scamming activities," IEEE transactions on visualization and computer graphics 25 (1), 225-234

[12] M Jakobsson, "Two-factor inauthentication?the rise in SMS phishing attacks" Computer Fraud & Security 2018 (6), 6-8

[13] M. Dhiman, M. Jakobsson, T.-F. Yen, "Breaking and fixing content-based filtering," 2017 APWG Symposium on Electronic Crime Research (eCrime), 52-56

[14] M. Jakobsson, "Addressing sophisticated email attacks," 2017 International Conference on Financial Cryptography and Data Security, 310-317

[15] M. Jakobsson, "User trust assessment: a new approach to combat deception," Proceedings of the 6th Workshop on Socio-Technical Aspects in Security and Trust, 2016, pages 73-78

[16] H. Siadati, T. Nguyen, P. Gupta, M. Jakobsson, "Mind your SMSes: Mitigating social engineering in second factor authentication," Computers and Security, 2016

APPENDIX A

IPR2025-00070
CrowdStrike EX1003 Page 182

[17] M. Jakobsson, W. Leddy, "Could you fall for a scam? Spam filters are passe. What we need is software that unmasks fraudsters," IEEE Spectrum 53 (5), 2016, 40-55

[18] N. Sae-Bae, M. Jakobsson, *Hand Authentication on Multi-Touch Tablets*, HotMobile 2014

[19] Y. Park, J. Jones, D. McCoy, E. Shi, M. Jakobsson, *Scambaiter: Understanding Targeted Nigerian Scams on Craigslist*, NDSS 2014

[20] D. Balfanz, R. Chow, O. Eisen, M. Jakobsson, S. Kirsch, S. Matsumoto, J. Molina, and P. van Oorschot, "The future of authentication," Security & Privacy, IEEE, 10(1), 22-27, 2012.

[21] M. Jakobsson, and H. Siadati, *Improved Visual Preference Authentication: Socio-Technical Aspects in Security and Trust*, (STAST), 2012 Workshop on IEEE, 27–34, 2012.

[22] M. Jakobsson, R. I. Chow, and J. Molina, "Authentication-Are We Doing Well Enough?[Guest Editors' Introduction]" Security & Privacy, IEEE, 10(1), 19-21, 2012.

[23] E. Shi, Y. Niu, M. Jakobsson, and R. Chow, "Implicit authentication through learning user behavior," Information Security, 99-113, Springer Berlin Heidelberg, 2011.

[24] M. Jakobsson and K. Johansson, "Practical and Secure Software-Based Attestation," Lightweight Security & Privacy: Devices, Protocols and Applications (LightSec), 1–9, 2011.

[25] A. Juels, D. Catalano, and M.Jakobsson, *Coercion-resistant electronic elections: Towards Trustworthy Elections*, 37–63, Springer Berlin Heidelberg, 2010.

[26] M. Jakobsson and F. Menczer, "Web Forms and Untraceable DDoS Attacks," in Network Security, Huang, S., MacCallum, D., and Du, D. Z., Eds.,77–95, Springer, 2010.

[27] R. Chow, M. Jakobsson, R. Masuoka, J. Molina, Y. Niu, E. Shi, and Z. Song, "Authentication in the Clouds: A Framework and its Application to Mobile Users," 2010.

[28] X. Wang, P. Golle, M. Jakobsson, and A. Tsow, "Deterring voluntary trace disclosure in re-encryption mix-networks," ACM Trans. Inf. Syst. Secur., 13(2), 1-24, 2010.

[29] X. Wang, P. Golle, M. Jakobsson, A.Tsow, "Deterring voluntary trace disclosure in re-encryption mix-networks," ACM Trans. Inf. Syst. Secur. 13(2): (2010)

[30] M. Jakobsson, and C. Soghoian, "Social Engineering in Phishing," Information Assurance, Security and Privacy Services, 4, 2009.

[31] M. Jakobsson, C. Soghoian and S. Stamm, "Phishing," Handbook of Financial Cryptography (CRC press, 2008)

[32] M. Jakobsson and A. Tsow, "Identity Theft," In John R. Vacca, Editor, "Computer And Information Security Handbook" (Morgan Kaufmann, 2008)

[33] S. Srikwan and M. Jakobsson, "Using Cartoons to Teach Internet Security," Cryptologia, vol. 32, no. 2, 2008

[34] M. Jakobsson, N. Johnson and P. Finn, "Why and How to Perform Fraud Experiments," IEEE Security and Privacy, March/April 2008 (Vol. 6, No. 2) pp. 66-68

[35] M. Jakobsson and S. Myers, "Delayed Password Disclosure," International Journal of Applied Cryptography, 2008, pp. 47-59.

[36] M. Jakobsson and S. Stamm, "Web Camouflage: Protecting Your Clients from Browser Sniffing Attacks," IEEE Security & Privacy Magazine. November/December 2007

[37] P. Finn and M. Jakobsson, "Designing and Conducting Phishing Experiments," IEEE Technology and Society Magazine, Special Issue on Usability and Security

[38] T. Jagatic, N. Johnson, M. Jakobsson and F. Menczer. "Social Phishing," The Communications of the ACM, October 2007

[39] A. Tsow, M. Jakobsson, L. Yang and S. Wetzel, "Warkitting: the Drive-by Subversion of Wireless Home Routers," Anti-Phishing and Online Fraud, Part II Journal of Digital Forensic Practice, Volume 1, Special Issue 3, November 2006

[40] M. Gandhi, M. Jakobsson and J. Ratkiewicz, "Badvertisements: Stealthy Click-Fraud with Unwitting Accessories," Anti-Phishing and Online Fraud, Part I Journal of Digital Forensic Practice, Volume 1, Special Issue 2, November 2006

[41] N. Ben Salem, J.-P. Hubaux and M. Jakobsson. "Reputation-based Wi-Fi Deployment," Mobile Computing and Communications Review, Volume 9, Number 3 (Best papers of WMASH 2004)

[42] N. Ben Salem, J. P. Hubaux, and M. Jakobsson. "Node Cooperation in Hybrid Ad hoc Networks," IEEE Transactions on Mobile Computing, Vol. 5, No. 4, April 2006.

[43] P. MacKenzie, T. Shrimpton, and M. Jakobsson. "Threshold Password-Authenticated Key Exchange," Journal of Cryptology, 2005

[44] A. Juels, M. Jakobsson, E. Shriver, and B. Hillyer. "How To Turn Loaded Dice Into Fair Coins." IEEE Transactions on Information Theory, vol. 46(3). May 2000. pp. 911–921.

[45] M. Jakobsson, P. MacKenzie, and J.P. Stern. "Secure and Lightweight Advertising on the Web," Journal of Computer Networks, vol. 31, issue 11–16, Elsevier North-Holland, Inc., 1999. pp. 1101–1109.

[46] M. Jakobsson, "Cryptographic Protocols," Chapter from The Handbook of Information Security. Hossein Bidgoli, Editor-in-Chief. Copyright John Wiley & Sons, Inc., 2005, Hoboken, N.J.

[47] M. Jakobsson, "Cryptographic Privacy Protection Techniques," Chapter from The Handbook of Information Security. Hossein Bidgoli, Editor-in-Chief. Copyright John Wiley & Sons, Inc., 2005, Hoboken, N.J.

[48] M. Jakobsson, E. Shi, P. Golle, R. Chow, "Implicit authentication for mobile devices," 4th USENIX Workshop on Hot Topics in Security (HotSec '09); 2009 August 11; Montreal, Canada.

[49] R. Chow, P. Golle, M. Jakobsson, E. Shi, J. Staddon, R. Masuoka, J. Molina, "Controlling data in the cloud: outsourcing computation without outsourcing control," Proceedings of the 2009 ACM Workshop on Cloud Computing Security (CCSW 2009); 2009 November 13; Chicago, IL. NY: ACM; 2009; pp. 85–90.

[50] M. Jakobsson, H. Siadati, M. Dhiman, "Liar Buyer Fraud, and How to Curb It," NDSS, 2015

[51] M. Jakobsson, T.-F. Yen, "How Vulnerable Are We To Scams?," BlackHat, 2015

[52] M. Jakobsson, "How to Wear Your Password," BlackHat, 2014

[53] M. Jakobsson and G. Stewart, "Mobile Malware: Why the Traditional AV Paradigm is Doomed, and How to Use Physics to Detect Undesirable Routines," in BlackHat, 2013.

[54] M. Jakobsson, and H. Siadati,"SpoofKiller: You Can Teach People How to Pay, but Not How to Pay Attention" in Socio-Technical Aspects in Security and Trust (STAST), 2012 Workshop on, 3-10, 2012.

[55] M. Jakobsson, and M. Dhiman,"The benefits of understanding passwords," in Proceedings of the 7th USENIX conference on Hot Topics in Security, Berkeley, CA, USA, 2012.

[56] M. Jakobsson, and S. Taveau, "The Case for Replacing Passwords with Biometrics," Mobile Security Technologies, 2012.

[57] M. Jakobsson and D. Liu, "Bootstrapping mobile PINs using passwords," W2SP, 2011.

[58] M. Jakobsson and R. Akavipat, "Rethinking passwords to adapt to constrained keyboards," 2011.

[59] Y. Niu, E. Shi, R. Chow, P. Golle, and M. Jakobsson, "One Experience Collecting Sensitive Mobile Data," In USER Workshop of SOUPS, 2010.

[60] E. Shi, Y. Niu, M. Jakobsson, and R. Chow, "Implicit Authentication through Learning User Behavior," 2010.

[61] M. Jakobsson and K. Johansson, Assured Detection of Malware With Applications to Mobile Platforms, 2010.

[62] M. Jakobsson and K. Johansson, "Retroactive Detection of Malware With Applications to Mobile Platforms," in HotSec 2010, Washington, DC, 2010.

[63] M. Jakobsson, A Central Nervous System for Automatically Detecting Malware, 2009.

[64] R. Chow, P. Golle, M. Jakobsson, R. Masuoka, J. Molina, E. Shi, and J. Staddon, "Controlling data in the cloud: outsourcing computation without outsourcing control," ACM workshop on Cloud computing security (CCSW), 2009.

[65] M. Jakobsson and A. Juels, "Server-Side Detection of Malware Infection," in New Security Paradigms Workshop (NSPW), Oxford, UK, 2009.

[66] M. Jakobsson, "Captcha-free throttling," Proceedings of the 2nd ACM workshop on Security and artificial intelligence, 15–22, 2009.

[67] M. Jakobsson, E. Shi, P. Golle, and R. Chow, "Implicit authentication for mobile devices," Proceedings of the 4th USENIX conference on Hot topics in security, 9–9, 2009.

[68] C. Soghoian, O. Friedrichs and M. Jakobsson, "The Threat of Political Phishing," International Symposium on Human Aspects of Information Security & Assurance (HAISA 2008)

[69] R. Chow, P. Golle, M. Jakobsson, L. Wang and X. Wang, "Making CAPTCHAs Clickable," In proc. of HotMobile 2008.

[70] M. Jakobsson, A. Juels, and J. Ratkiewicz, "Privacy-Preserving History Mining for Web Browsers," Web 2.0 Security and Privacy, 2008.

[71] M. Jakobsson, E. Stolterman, S. Wetzel, L. Yang, "Love and Authentication," (Notes) ACM Computer/Human Interaction Conference (CHI), 2008. Also see `www.I-forgot-my-password.com`

[72] M. Jakobsson and S. Myers, "Delayed Password Disclosure," Proceedings of the 2007 ACM workshop on Digital Identity Management

[73] M. Jakobsson, S. Stamm, Z. Ramzan, "JavaScript Breaks Free," W2SP '07

APPENDIX A

IPR2025-00070
CrowdStrike EX1003 Page 186

[74] A. Juels, S. Stamm, M. Jakobsson, "Combatting Click Fraud via Premium Clicks," USENIX Security 2007

[75] R. Chow, P. Golle, M. Jakobsson, X. Wang, "Clickable CAPTCHAs," Ad-Fraud '07 Workshop; 2007 September 14; Stanford, CA, USA

[76] S. Stamm, Z. Ramzan, and M. Jakobsson, "Drive-by Pharming," In Proceedings of Information and Communications Security, 9th International Conference, ICICS 2007

[77] M. Jakobsson, A. Tsow, A. Shah, E. Blevis, Y.-K. Lim, "What Instills Trust? A Qualitative Study of Phishing," USEC '07.

[78] R. Akavipat, V. Anandpara, A. Dingman, C. Liu, D. Liu, K. Pongsanon, H. Roinestad and M. Jakobsson, "Phishing IQ Tests Measure Fear, not Ability," USEC '07.

[79] M. Jakobsson, "The Human Factor in Phishing," American Conference Institute's Forum on Privacy & Security of Consumer Information, 2007

[80] S. Srikwan, M. Jakobsson, A. Albrecht and M. Dalkilic, "Trust Establishment in Data Sharing: An Incentive Model for Biodiversity Information Systems," TrustCol 2006

[81] J.Y. Choi, P. Golle, M. Jakobsson, "Tamper-Evident Digital Signatures: Protecting Certification Authorities Against Malware," DACS '06

[82] L. Yang, M. Jakobsson, S. Wetzel, "Discount Anonymous On Demand Routing for Mobile Ad hoc Networks," SECURECOMM '06

[83] P. Golle, X. Wang, M. Jakobsson, A. Tsow, "Deterring Voluntary Trace Disclosure in Re-encryption Mix Networks." IEEE S&P '06

[84] M. Jakobsson, A. Juels, T. Jagatic, "Cache Cookies for Browser Authentication (Extended Abstract)," IEEE S&P '06

[85] M. Jakobsson and J. Ratkiewicz, "Designing Ethical Phishing Experiments: A study of (ROT13) rOnl auction query features.", WWW '06

[86] M. Jakobsson and S. Stamm. "Invasive Browser Sniffing and Countermeasures," WWW '06

[87] J.Y. Choi, P. Golle and M. Jakobsson. "Auditable Privacy: On Tamper-Evident Mix Networks," Financial Crypto '06

[88] A. Juels, D. Catalano and M. Jakobsson. "Coercion-Resistant Electronic Elections," WPES '05

[89] V. Griffith and M. Jakobsson. "Messin' with Texas, Deriving Mother's Maiden Names Using Public Records," ACNS '05, 2005.

[90] M. Jakobsson and L. Yang. "Quantifying Security in Hybrid Cellular Networks," ACNS '05, 2005

[91] Y.-C. Hu, M. Jakobsson, and A. Perrig. "Efficient Constructions for One-way Hash Chains," ACNS '05, 2005

[92] M. Jakobsson. "Modeling and Preventing Phishing Attacks," Phishing Panel in Financial Cryptography '05. 2005, abstract in proceedings.

[93] N. Ben Salem, J.-P. Hubaux, and M. Jakobsson. "Reputation-based Wi-Fi Deployment Protocols and Security Analysis," In WMASH '04. ACM Press, 2004. pp. 29–40.

[94] M. Jakobsson and S. Wetzel. "Efficient Attribute Authentication with Applications to Ad Hoc Networks," In VANET '04. ACM Press, 2004. pp. 38–46.

[95] M. Jakobsson, X. Wang, and S. Wetzel. "Stealth Attacks in Vehicular Technologies," Invited paper. In Proceedings of IEEE Vehicular Technology Conference 2004 Fall (VTC-Fall 2004). IEEE, 2004.

[96] A. Ambainis, H. Lipmaa, and M. Jakobsson. "Cryptographic Randomized Response Technique," In PKC '04. LNCS 2947. Springer-Verlag, 2004. pp. 425–438.

[97] P. Golle, M. Jakobsson, A. Juels, and P. Syverson. "Universal Re-encryption for Mixnets," In CT-RSA '04. LNCS 2964. Springer-Verlag, 2004. pp. 163–178.

[98] P. Golle and M. Jakobsson. "Reusable Anonymous Return Channels," In WPES '03. ACM Press, 2003. pp. 94–100.

[99] M. Jakobsson, S. Wetzel, B. Yener. "Stealth Attacks on Ad-Hoc Wireless Networks," In IEEE VTC '03, 2003.

[100] N. Ben Salem, L. Buttyan, J.-P. Hubaux, and M. Jakobsson. "A Charging and Rewarding Scheme for Packet Forwarding in Multi-hop Cellular Networks," In ACM MobiHoc '03. ACM Press, 2003. pp. 13–24.

[101] M. Jakobsson, J.-P.Hubaux and L. Buttyan. "A Micro-Payment Scheme Encouraging Collaboration in Multi-Hop Cellular Networks," In FC '03. LNCS 2742. Springer-Verlag, 2003. pp. 15–33.

[102] M. Jakobsson, T. Leighton, S. Micali and M. Szydlo. "Fractal Merkle Tree Representation and Traversal," In RSA-CT '03 2003.

[103] A. Boldyreva and M Jakobsson. "Theft protected proprietary certificates," In DRM '02. LNCS 2696, 2002. pp. 208–220.

APPENDIX A

IPR2025-00070
CrowdStrike EX1003 Page 188

[104] P. Golle, S. Zhong, M. Jakobsson, A. Juels, and D. Boneh. "Optimistic Mixing for Exit-Polls," In Asiacrypt '02. LNCS 2501. Springer-Verlag, 2002. pp. 451–465.

[105] P. MacKenzie, T. Shrimpton, and M. Jakobsson. "Threshold Password-Authenticated Key Exchange," In CRYPTO '02. LNCS 2442. Springer-Verlag, 2002. pp. 385–400.

[106] M. Jakobsson. "Fractal Hash Sequence Representation and Traversal," In Proceedings of the 2002 IEEE International Symposium on Information Theory (ISIT '02). 2002. pp. 437–444.

[107] M. Jakobsson, A. Juels, and R. Rivest. "Making Mix Nets Robust For Electronic Voting By Randomized Partial Checking," In Proceedings of the 11th USENIX Security Symposium. USENIX Association, 2002. pp. 339–353.

[108] D. Coppersmith and M. Jakobsson. "Almost Optimal Hash Sequence Traversal," In Financial Crypto '02. 2002.

[109] M. Jakobsson. "Financial Instruments in Recommendation Mechanisms," In Financial Crypto '02. 2002.

[110] J. Garay, and M. Jakobsson. "Timed Release of Standard Digital Signatures," In Financial Crypto '02. 2002.

[111] F. Menczer, N. Street, N. Vishwakarma, A. Monge, and M. Jakobsson. "Intellishopper: A Proactive, Personal, Private Shopping Assistant," In AAMAS '02. ACM Press, 2002. pp. 1001–1008.

[112] M. Jakobsson, A. Juels, and P. Nguyen. "Proprietary Certificates," In CT-RSA '02. LNCS 2271. Springer-Verlag, 2002. pp. 164–181.

[113] M. Jakobsson and A. Juels. "An Optimally Robust Hybrid Mix Network," In PODC '01. ACM Press. 2001. pp. 284–292.

[114] M. Jakobsson and M. Reiter. "Discouraging Software Piracy Using Software Aging," In DRM '01. LNCS 2320. Springer-Verlag, 2002. pp. 1–12.

[115] M. Jakobsson and S. Wetzel. "Security Weaknesses in Bluetooth," In CT–RSA '01. LNCS 2020. Springer-Verlag, 2001. pp. 176–191.

[116] M. Jakobsson and D. Pointcheval. "Mutual Authentication for Low-Power Mobile Devices," In Financial Crypto '01. LNCS 2339. Springer-Verlag, 2001. pp. 178–195.

[117] M. Jakobsson, D. Pointcheval, and A. Young. "Secure Mobile Gambling," In CT–RSA '01. LNCS 2020. Springer-Verlag, 2001. pp. 110–125.

[118] M. Jakobsson and S.Wetzel. "Secure Server-Aided Signature Generation," In PKC '01. LNCS 1992. Springer-Verlag, 2001. pp. 383–401.

APPENDIX A

IPR2025-00070
CrowdStrike EX1003 Page 189

[119] M. Jakobsson and A. Juels. "Addition of ElGamal Plaintexts," In T. Okamoto, ed., ASIACRYPT '00. LNCS 1976. Springer-Verlag, 2000. pp. 346–358.

[120] M. Jakobsson, and A. Juels. "Mix and Match: Secure Function Evaluation via Ciphertexts," In ASIACRYPT '00. LNCS 1976. Springer-Verlag, 2000. pp. 162–177.

[121] R. Arlein, B. Jai, M. Jakobsson, F. Monrose, and M. Reiter. "Privacy-Preserving Global Customization," In ACM E-Commerce '00. ACM Press, 2000. pp. 176–184.

[122] C.-P. Schnorr and M. Jakobsson. "Security of Signed ElGamal Encryption," In ASIACRYPT '00. LNCS 1976. Springer-Verlag, 2000. pp. 73–89.

[123] P. Bohannon, M. Jakobsson, and S. Srikwan. "Cryptographic Approaches to Privacy in Forensic DNA Databases," In Public Key Cryptography '00. LNCS 1751. Springer-Verlag, 2000, pp. 373–390.

[124] J. Garay, M. Jakobsson, and P. MacKenzie. "Abuse-free Optimistic Contract Signing," In CRYPTO '99. LNCS 1666. Springer-Verlag, 1999. pp. 449–466.

[125] M. Jakobsson. "Flash Mixing," In PODC '99. ACM Press, 1999. pp. 83–89.

[126] G. Di Crescenzo, N. Ferguson, R. Impagliazzo, and M. Jakobsson. "How To Forget a Secret," In STACS '99. LNCS 1563. Springer-Verlag, 1999. pp. 500–509.

[127] M. Jakobsson, D. M'Raihi, Y. Tsiounis, and M. Yung. "Electronic Payments: Where Do We Go from Here?," In CQRE (Secure) '99. LNCS 1740. Springer-Verlag, 1999. pp. 43–63.

[128] C.P. Schnorr and M. Jakobsson. "Security Of Discrete Log Cryptosystems in the Random Oracle + Generic Model," In Conference on The Mathematics of Public-Key Cryptography. 1999.

[129] M. Jakobsson and A. Juels "Proofs of Work and Breadpudding Protocols," In CMS '99. IFIP Conference Proceedings, Vol. 152. Kluwer, B.V., 1999. pp. 252 – 272.

[130] M. Jakobsson and C-P Schnorr. "Efficient Oblivious Proofs of Correct Exponentiation," In CMS '99. IFIP Conference Proceedings, Vol. 152. Kluwer, B.V., 1999. pp. 71–86.

[131] M. Jakobsson, P. MacKenzie, and J.P. Stern. "Secure and Lightweight Advertising on the Web," In World Wide Web '99

APPENDIX A

IPR2025-00070
CrowdStrike EX1003 Page 190

[132] M. Jakobsson, J.P. Stern, and M. Yung. "Scramble All, Encrypt Small," In Fast Software Encryption '99. LNCS 1636. Springer-Verlag, 1999. pp. 95–111.

[133] M. Jakobsson and J. Mueller. "Improved Magic Ink Signatures Using Hints," In Financial Cryptography '99. LNCS 1648. Springer-Verlag, 1999. pp. 253–268.

[134] M. Jakobsson. "Mini-Cash: A Minimalistic Approach to E-Commerce," In Public Key Cryptography '99. LNCS 1560. Springer-Verlag, 1999. pp. 122–135.

[135] M. Jakobsson. "On Quorum Controlled Asymmetric Proxy Re-encryption," In Public Key Cryptography '99. LNCS 1560. Springer-Verlag, 1999. pp. 112–121.

[136] M. Jakobsson and A. Juels. "X-Cash: Executable Digital Cash," In Financial Cryptography '98. LNCS 1465. Springer-Verlag, 1998. pp. 16–27.

[137] M. Jakobsson and D. M'Raihi. "Mix-based Electronic Payments," In Proceedings of the Selected Areas in Cryptography. LNCS 1556. Springer-Verlag, 1998. pp. 157173.

[138] M. Jakobsson, E. Shriver, B. Hillyer, and A. Juels. "A Practical Secure Physical Random Bit Generator," In CCS '98: Proceedings of the 5th ACM conference on Computer and communications security. ACM Press, 1998. pp. 103–111.

[139] M. Jakobsson. "A Practical Mix," In Advances in Cryptology – EuroCrypt '98. LNCS 1403. Springer-Verlag, 1998. pp. 448–461.

[140] M. Jakobsson and M. Yung. "On Assurance Structures for WWW Commerce," In Financial Cryptography '98. LNCS 1465. Springer-Verlag, 1998. pp. 141–157.

[141] E. Gabber, M. Jakobsson, Y. Matias, and A. Mayer. "Curbing Junk E-Mail via Secure Classification," In Financial Cryptography '98. LNCS 1465. Springer-Verlag, 1998. pp. 198–213.

[142] M. Jakobsson and M. Yung. "Distributed 'Magic Ink' Signatures," In Advances in Cryptology – EuroCrypt '97. LNCS 1233. Springer-Verlag, 1997. pp. 450–464.

[143] M. Jakobsson and M. Yung. "Applying Anti-Trust Policies to Increase Trust in a Versatile E-Money System," In Financial Cryptography '97. LNCS 1318. Springer-Verlag, 1997. pp. 217–238.

[144] A. Herzberg, M. Jakobsson, S. Jarecki, H. Krawczyk, and M. Yung. "Proactive public-key and signature schemes," In Proceedings of the 4th Annual Conference on Computer Communications Security. ACM Press, 1997. pp. 100–110.

APPENDIX A

IPR2025-00070
CrowdStrike EX1003 Page 191

[145] M. Bellare, M. Jakobsson, and M. Yung. "Round-Optimal Zero-Knowledge Arguments Based on any One-Way Function," In Advances in Cryptology – EuroCrypt '97. LNCS 1233. Springer-Verlag, 1997. pp. 280–305.

[146] M. Jakobsson and M. Yung. "Proving Without Knowing," In Crypto '96. LNCS 1109. Springer-Verlag, 1996. pp. 186–200.

[147] M. Jakobsson, K. Sako, and R. Impagliazzo. "Designated Verifier Proofs and Their Applications," In Advances in Cryptology – EuroCrypt '96. LNCS 1070. Springer-Verlag, 1996. pp. 143–154.

[148] M. Jakobsson and M. Yung. "Revokable and Versatile Electronic Money," In CCS '96: Proceedings of the 3rd ACM conference on Computer and communications security. ACM Press, 1996. pp. 76–87.

[149] M. Jakobsson. "Ripping Coins for a Fair Exchange," In Advances in Cryptology – EuroCrypt '95. LNCS 921. Springer-Verlag, 1995. pp. 220–230.

[150] M. Jakobsson. "Blackmailing using Undeniable Signatures," In Advances in Cryptology EuroCrypt '94. LNCS 950. Springer-Verlag, 1994. pp. 425–427.

[151] M. Jakobsson. "Reducing costs in identification protocols," Crypto '92, 1992.

[152] M. Jakobsson. "Machine-Generated Music with Themes," In International Conference on Artificial Neural Networks '92. Vol 2. Amsterdam: Elsevier, 1992. pp. 1645–1646

[153] M. Jakobsson, "Social Engineering 2.0: What's Next," McAfee Security Journal, Fall 2008

[154] M. Jakobsson and S. Myers, "Delayed Password Disclosure," ACM SIGACT News archive, Volume 38, Issue 3 (September 2007), pp. 56 - 75

[155] V. Griffith and M. Jakobsson. "Messin' with Texas, Deriving Mother's Maiden Names Using Public Records," CryptoBytes, 2007.

[156] M. Jakobsson, A. Juels, J. Ratkiewicz, "Remote-Harm Detection," Beta-version available at `rhd.ravenwhitedevelopment.com/`

[157] S. Stamm, M. Jakobsson, "Social Malware," Experimental results available at `www.indiana.edu/ phishing/verybigad/`

[158] M. Jakobsson. "Privacy vs. Authenticity," Ph.D. Thesis, University of California at San Diego, 1997

[159] Markus Jakobsson, Automatic PIN creation using passwords, US20130125214 A1, 2012.

[160] Markus Jakobsson, Systems and methods for creating a user credential and authentication using the created user credential, US 20130111571 A1, 2012.

[161] Markus Jakobsson, Password check by decomposing password, US 20120284783 A1, 2012.

[162] Markus Jakobsson, William Leddy, System and methods for protecting users from malicious content, US 20120192277 A1, 2011.

[163] Markus Jakobsson, Karl-Anders R. Johansson, Auditing a device, US8370935 B1, 2011.

[164] Markus Jakobsson, Methods and Apparatus for Efficient Computation of One-Way Chains in Cryptographic Applications, US20120303969 A1, 2011.

[165] Markus Jakobsson, Automatic PIN creation using password, US 20120110634 A1, 2011.

[166] Markus Jakobsson, Jim Roy Palmer, Gustavo Maldonado, Interactive CAPTCHA, US20130007875 A1, 2011.

[167] Markus Jakobsson, System access determination based on classification of stimuli, US 20110314559 A1, 2011.

[168] Markus Jakobsson, System access determination based on classification of stimuli, WO 2011159356 A1, 2011.

[169] Markus Jakobsson, Method, medium, and system for reducing fraud by increasing guilt during a purchase transaction, US8458041 B1, 2011.

[170] Markus Jakobsson, Visualization of Access Information, US 20120233314 A1, 2011.

[171] Markus Jakobsson, Richard Chow,Runting Shi, Implicit authentication, US20120137340 A1, 2010.

[172] Markus Jakobsson, Karl-Anders R. Johansson, Auditing a device, EP2467793 A1, 2010.

[173] Markus Jakobsson, Event log authentication using secure components, US 20110314297 A1, 2010.

[174] Markus Jakobsson, Philippe J.P. Golle, Risk-based alerts, US 20110314426 A1, 2010.

[175] Markus Jakobsson, Karl-Anders R. Johansson, Auditing a device, US 20110041178 A1, 2010.

[176] M. Jakobsson, A. Juels, J. Kaliski Jr, S. Burton and others, Identity authentication system and method, US Patent 7,502,933, 2009.

[177] Markus Jakobsson, Method and system for facilitating throttling of interpolation-based authentication, US8219810 B2, 2009.

[178] Markus Jakobsson, Karl-Anders R. Johansson, Auditing a device, US8375442 B2, 2009.

[179] Markus Jakobsson, Karl-Anders R. Johansson, Auditing a device, US 20110041180 A1, 2009.

[180] Markus Jakobsson, Pattern-based application classification, US 20110055925 A1, 2009.

[181] Markus Jakobsson, Method and apparatus for detecting cyber threats, US8286225 B2, 2009.

[182] Markus Jakobsson, Method and apparatus for detecting cyber threats, US20110035784 A1, 2009.

[183] Markus Jakobsson, CAPTCHA-free throttling, US8312073 B2, 2009.

[184] Markus Jakobsson, Captcha-free throttling, US 20110035505 A1, 2009.

[185] Markus Jakobsson, Christopher Soghoian, Method and apparatus for throttling access using small payments, US 20100153275 A1, 2008.

[186] Markus Jakobsson, Christopher Soghoian, Method and apparatus for mutual authentication using small payments, US 20100153274 A1, 2008.

[187] Philippe J.P. Golle, Markus Jakobsson,Richard Chow, Resetting a forgotten password using the password itself as authentication, US 20100125906 A1, 2008.

[188] Philippe J. P. Golle, Markus Jakobsson,Richard Chow, Authenticating users with memorable personal questions, US8161534 B2, 2008.

[189] Richard Chow, Philippe J.P. Golle, Markus Jakobsson, Jessica N. Staddon, Authentication based on user behavior, US20100122329 A1, 2008.

[190] Richard Chow, Philippe J.P. Golle, Markus Jakobsson, Jessica N. Staddon, Enterprise password reset, US 20100122340 A1, 2008.

[191] Markus Jakobsson, Methods and apparatus for efficient computation of one-way chains in cryptographic applications, US8086866 B2, 2008.

[192] Richard Chow, Philippe J. P. Golle, Markus Jakobsson, Selectable captchas, US8307407 B2, 2008.

[193] Markus Jakobsson, Ari Juels, Sidney Louis Stamm, Method and apparatus for combatting click fraud, US 20080162227 A1, 2007.

[194] Jakobsson, Method and apparatus for evaluating actions performed on a client device, US 20080037791 A1, 2007.

[195] Jakobsson, Ari Juels, Method and apparatus for storing information in a browser storage area of a client device, US 20070106748 A1, 2006.

[196] Markus Jakobsson, Steven Andrew Myers, Anti-phising logon authentication object oriented system and method, WO 2006062838 A1, 2005.

[197] Jakobsson, Jean-Pierre Hubaux,Levente Buttyan, Micro-payment scheme encouraging collaboration in multi-hop cellular networks, US 20050165696 A1, 2004.

[198] Andrew Nanopoulos, Karl Ackerman, Piers Bowness, William Duane, Markus Jakobsson, Burt Kaliski, Dmitri Pal, Shane D. Rice,Less , System and method providing disconnected authentication, WO2005029746 A3, 2004.

[199] Andrew Nanopoulos, Karl Ackerman, Piers Bowness, William Duane, Markus Jakobsson, Burt Kaliski, Dmitri Pal, Shane Rice, Ronald Rivest, Less , System and method providing disconnected authentication, US 20050166263 A1, 2004.

[200] Andrew Nanopoulos, Karl Ackerman, Piers Bowness, William Duane, Markus Jakobsson, Burt Kaliski, Dmitri Pal, Shane D. Rice,Less , Systeme et procede d'authentification deconnectee, WO 2005029746 A2, 2004.

[201] Markus Jakobsson, Ari Juels, Burton S. Kaliski Jr., Identity authentication system and method, WO2004051585 A3, 2003.

[202] Markus Jakobsson, Ari Juels, Burton S. Kaliski, Jr., Identity authentication system and method, US 7502933 B2, 2003.

[203] Markus Jakobsson, Ari Juels, Burton S. Kaliski Jr., Systeme et procede de validation d'identite, WO 2004051585 A2, 2003.

[204] Markus Jakobsson, Burton S. Kaliski, Jr., Method and apparatus for graph-based partition of cryptographic functionality, US7730518 B2, 2003.

[205] Markus Jakobsson, Phong Q. Nguyen, Methods and apparatus for private certificates in public key cryptography, US7404078 B2, 2002.

[206] Jakobsson, Philip MacKenzie, Thomas Shrimpton, Method and apparatus for performing multi-server threshold password-authenticated key exchange, US20030221102 A1, 2002.

[207] Markus Jakobsson, Philip D MacKenzie, Method and apparatus for distributing shares of a password for use in multi-server password authentication, US 7073068 B2, 2002.

[208] Markus Jakobsson, Methods and apparatus for efficient computation of one-way chains in cryptographic applications, EP 1389376 A1 (text from WO2002084944A1), 2002.

[209] Markus Jakobsson, Adam Lucas Young, Method and apparatus for identification tagging documents in a computer system, US 7356845 B2, 2002.

[210] Juan A. Garay, Markus Jakobsson, Methods and apparatus for computationally-efficient generation of secure digital signatures, US 7366911 B2, 2001.

[211] Markus Jakobsson, Methods and apparatus for efficient computation of one-way chains in cryptographic applications, US 7404080 B2, 2001.

[212] Markus Jakobsson, Susanne Gudrun Wetzel, Securing the identity of a bluetooth master device (bd addr) against eavesdropping by preventing the association of a detected channel access code (cac) with the identity of a particular bluetooth device, WO2002019641 A3, 2001.

[213] Markus Jakobsson, Susanne Gudrun Wetzel, Procede et appareil permettant d'assurer la securite d'utilisateurs de dispositifs a capacites bluetooth, WO 2002019641 A2, 2001.

[214] Robert M. Arlein, Ben Jai, Markus Jakobsson, Fabian Monrose, Michael Kendrick Reiter, Less , Methods and apparatus for providing privacy-preserving global customization, US 7107269 B2, 2001.

[215] Markus Jakobsson, Susanne Gudrun Wetzel, Secure distributed computation in cryptographic applications, US6950937 B2, 2001.

[216] Markus Jakobsson, Susanne Gudrun Wetzel, Method and apparatus for ensuring security of users of short range wireless enable devices, US6981157 B2, 2001.

[217] Markus Jakobsson, Susanne Gudrun Wetzel, Method and apparatus for ensuring security of users of bluetooth TM-enabled devices, US 6574455 B2, 2001.

[218] Garay; Juan A. (West New York, NJ), Jakobsson; M. (Hoboken, NJ), Kristol; David M. (Summit, NJ), Mizikovsky; Semyon B.(Morganville, NJ), Cryptographic key processing and storage , 7023998, 2001.

[219] Markus Jakobsson, Method, apparatus, and article of manufacture for generating secure recommendations from market-based financial instrument prices, US 6970839 B2, 2001.

[220] Markus Jakobsson, Encryption method and apparatus with escrow guarantees, US7035403 B2, 2001.

[221] Jakobsson, Call originator access control through user-specified pricing mechanism in a communication network, US20020099670 A1, 2001.

[222] Markus Jakobsson, Claus Peter Schnorr, Tagged private information retrieval, US7013295 B2, 2000.

[223] Markus Jakobsson, Secure enclosure for key exchange, US 7065655 B1, 2000.

[224] Markus Jakobsson, Michael Kendrick Reiter, Software aging method and apparatus for discouraging software piracy, US 7003110 B1, 2000.

[225] Markus Jakobsson, Probabilistic theft deterrence, US6501380 B1, 2000.

[226] Markus Jakobsson, Michael Kendrick Reiter, Abraham Silberschatz, Anonymous and secure electronic commerce, EP 1150227 A1, 2000.

[227] Markus Jakobsson, Ari Juels, Proofs of work and bread pudding protocols, US 7356696 B1, 2000.

[228] Markus Jakobsson, Joy Colette Mueller, Methods of protecting against spam electronic mail, US7644274 B1, 2000.

[229] Philip L. Bohannon, Markus Jakobsson,Fabian Monrose, Michael Kendrick Reiter,Susanne Gudrun Wetzel, Less , Generation of repeatable cryptographic key based on varying parameters, EP1043862 B1, 2000.

[230] Markus Jakobsson, Ari Juels, Mix and match: a new approach to secure multiparty computation, US6772339 B1, 2000.

[231] Markus Jakobsson, Ari Juels, Mixing in small batches, US6813354 B1, 2000.

[232] Philip L. Bohannon, Markus Jakobsson,Fabian Monrose, Michael Kendrick Reiter,Susanne Gudrun Wetzel, Less , Generation of repeatable cryptographic key based on varying parameters, US 6901145 B1, 36566

[233] Markus Jakobsson, Claus Peter Schnorr, Non malleable encryption method and apparatus using key-encryption keys and digital signature, US6931126 B1, 2000.

[234] Markus Jakobsson, Claus Peter Schnorr, Non malleable encryption method and apparatus using key-encryption keys and digital signature, US 6931126 B1, 2000.

[235] Markus Jakobsson, Flash mixing apparatus and method, US 6598163 B1, 1999.

[236] Markus Jakobsson, Minimalistic electronic commerce system, US 6529884 B1, 1999.

[237] Markus Jakobsson, Method and system for providing translation certificates, US 6687822 B1, 1999.

[238] Markus Jakobsson, Verification of correct exponentiation or other operations in cryptographic applications, US6978372 B1, 1999.

[239] Markus Jakobsson, Non malleable encryption apparatus and method, US 6507656 B1, 1999.

[240] Markus Jakobsson, Method and system for quorum controlled asymmetric proxy encryption, US 6587946 B1, 1998.

[241] Markus Jakobsson, Practical mix-based election scheme, US 6317833 B1, 1998.

[242] Markus Jakobsson, Ari Juels, Method and apparatus for extracting unbiased random bits from a potentially biased source of randomness, US 6393447 B1, 1998.

[243] Markus Jakobsson, Ari Juels, Executable digital cash for electronic commerce, US6157920 A, 1998.

[244] Bruce Kenneth Hillyer, Markus Jakobsson, Elizabeth Shriver, Storage device random bit generator, US 6317499 B1, 1998.

[245] Markus Jakobsson, Method and apparatus for encrypting, decrypting, and providing privacy for data values, US 6049613 A, 1998.