

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

CROWDSTRIKE, INC.
Petitioner

v.

WEBROOT INC.
Patent Owner

Case No. IPR2023-01053
Patent No. 11,409,869

DECLARATION OF DR. MARKUS JAKOBSSON, PH.D.

TABLE OF CONTENTS

I.	Introduction.....	4
A.	Materials Reviewed.....	4
B.	Background and Qualifications.....	6
II.	Legal Framework.....	15
III.	Opinion	22
A.	Overview of the '869 Patent	22
B.	Overview of the Relevant File Histories.....	26
1.	Overview of the '844 Parent Patent's File History.....	27
2.	Overview of the '869 Patent's File History	29
C.	Level of a Person Having Ordinary Skill in the Art	30
D.	Claim Construction	31
E.	Summary of the Prior Art References.....	32
1.	Kumar	32
2.	Tahan.....	36
F.	Opinions Regarding Ground 1: Claims 1-59 are rendered obvious over <i>Kumar</i> or, Alternatively, <i>Kumar</i> in view of <i>Tahan</i>	39
1.	Claim 1	39
2.	Claim 2 – The computer-implemented method according to claim 1, wherein the plurality of static data points are extracted without decrypting or unpacking the executable file.	76
3.	Claim 3 – The computer-implemented method according to claim 1, wherein the one or more support vector machines builds the probabilistic model by performing data analysis and pattern recognition on the one or more feature vectors.	79
4.	Claim 5 – The computer-implemented method according to claim 1, wherein the executable file is identified in response to a detected condition.	81
5.	Claim 6 – The computer-implemented method according to claim 5, wherein the detected condition is user request for a file download....	83
6.	Claim 7 – The computer-implemented method according to claim 5, wherein the detected condition is the detection of a new file attempting to execute.	86

7. Claim 8 – The computer-implemented method according to claim 1, wherein the plurality of static data points represent predefined character strings in the executable file.....	87
8. Claim 9 – The computer-implemented method according to claim 1, wherein a determination of whether the executable file is harmful is used to retrain the classifier.	90
9. Claim 18(f) – evaluating the feature vector using a machine learning model to determine whether the executable file fits into one of the categories of files.	92
10.Claim 20 – The computer-implemented method according to claim 18, wherein the machine learning model comprises an artificial neural network.	94
11.Claim 22 – The computer-implemented method according to claim 18, wherein the machine learning model comprises a machine learning decision tree.	95
12.Claim 23 – The computer-implemented method according to claim 18, wherein the machine learning model comprises a Bayesian network.	95
13.Claim 24 – The computer-implemented method according to claim 18, wherein evaluating the feature vector using the machine learning model comprises evaluating the feature vector using support vector processing.	96
14.Claim 25 – The computer-implemented method according to claim 18, wherein the machine learning model comprises a clustering model.	97
15.Claim 31 – The computer-implemented method according to claim 18, further comprising preventing execution of the executable file based on classifying the executable file as harmful.	100
16.Claim 46(e) – at least one processor configured to perform operation of:	100
IV. Conclusion	102

I, Dr. Markus Jakobsson, declare the following:

I. INTRODUCTION

1. I have been retained by counsel for Petitioner as a technical expert in the above-captioned case. Specifically, I have been asked to render certain opinions in regard to the IPR petition with respect to U.S. Patent No. 11,409,869 (the “’869 Patent”) (Ex. 1001). I understand that the Challenged Claims are claims 1-59 and my opinions are limited to those claims. My compensation in this matter is not based on the substance of my opinions or the outcome of this matter. I have no financial interest in Petitioner.

A. Materials Reviewed

2. In reaching my opinions in this matter, I have reviewed the following materials:

- Exhibit 1001 – U.S. Patent No. 11,409,869 (“’869 *Patent*”)
- Exhibit 1002 – File History of U.S. Patent No. 11,409,869 (“’869 *FH*”)
- Exhibit 1004 – U.S. Patent No. 2012/0317644 to Kumar et al. (“*Kumar*”)
- Exhibit 1005 – Mal-ID: Automatic Malware Detection Using Common Segment Analysis and Meta-Features by Gil Tahan et al. (“*Tahan*”)
- Exhibit 1006 – File History of U.S. Patent No. 10,599,844 (“’844 *FH*”)
- Exhibit 1007 – 22-cv-00243, Dkt. 304, Plaintiffs’ Responsive Claim Construction Brief (“*Responsive Markman Brief, Dkt. 304*”)
- Exhibit 1008 – Declaration of June Ann Munford (“*Munford Dec.*”)

- Exhibit 1009 – 22-cv-00243, Dkt. 98, Plaintiffs’ Responsive Claim Construction Brief (“*Responsive Markman Brief, Dkt. 98*”)
- Exhibit 1010 – 22-cv-00243, Dkt. 236 Claim Construction Order (“*Markman Order*”)
- Exhibit 1015 – Declaration of Dr. Ingrid Hsieg Yee (“*Dr. Yee Dec.*”)
- Exhibit 1016 – U.S. Patent Application No. 2013/0326625 to Anderson et al. (“*Anderson*”)
- Exhibit 1017 – U.S. Patent No. 8,266,698 (“*Seshardi*”)
- Exhibit 1018 – U.S. Patent No. 9,489,514 to Mankin et al. (“*Mankin*”)
- Exhibit 1019 – Intrusion Detection Using The Support Vector Machine Enhanced With A Feature--Weight Kernel by Songlun Zhao et al. (“*Zhao*”)
- Exhibit 1020 – Intrusion Detection Using Neural Networks and Support Vector Machines by Srinivas Mukkamala et al. (“*Mukkamala*”)
- Exhibit 1021 – Classification of Malicious Domain Names using Support Vector Machine and Bi-gram Method by Nhauo Davuth et al. (“*Davuth*”)
- Exhibit 1022 – Malware behavioral detection and vaccine development by using a support vector model classifier by Ping Wang et al. (“*Wang*”)
- Exhibit 1023 – FIRMA: Malware Clustering and Network Signature Generation with Mixed Network Behaviors by M. Zubair Rafique (“*FIRMA*”)
- Exhibit 1024 – U.S. Patent Publication No. 2008/0319932 to Wen-tau Yih et al. (“*Yih*”)
- Exhibit 1025 – Comparative Analysis of Combinations of Dimension Reduction and Data Mining Techniques for Malware Detection, by Proceso L. Fernandez Jr. et al. (“*Fernandez*”)
- Exhibit 1026 – U.S. Patent No. 8,418,249 to Nucci et al. (“*Nucci*”)
- Exhibit 1027 – U.S. Patent No. 8,161,548 to Wan (“*Wan*”)

- Exhibit 1028 – U.S. Patent No. 8,875,289 B2 to Mahaffey et al. (“*Mahaffey*”)
- Exhibit 1029 – U.S. Patent No. 9,349,103 to Eberhardt, III et al. (“*Eberhardt*”)
- Exhibit 1030 – U.S. Patent No. 9,043,894 to Dennison et al. (“*Dennison*”)
- Exhibit 1031 – U.S. Patent No. 9,306,971 to Altman et al. (“*Altman*”)
- Exhibit 1032 – U.S. Patent Publication No. 2007/0245520 to Yong et al. (“*Yong*”)
- Exhibit 1033 – U.S. Patent No. 9,721,212 to Gupta et al. (“*Gupta*”)
- Exhibit 1034 – U.S. Patent No. 9,324,034 to Gupta et al. (“*Gupta II*”)
- Exhibit 1035 – U.S. Patent No. 10,783,254 to Sharma et al. (“*Sharma*”)
- Exhibit 1036 – “Ensemble Methods for Malware Diagnosis Based on One-class SVMs” by Xing An (“*Xing*”)
- Exhibit 1037 – U.S. Patent No. 9,306,966 to Eskin et al. (“*Eskin*”)

B. Background and Qualifications

3. I have summarized in this section my educational background, career history, and other qualifications relevant to this matter. I have also included a current version of my curriculum vitae, which is attached as Appendix A.

4. 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. Whereas the term “cloud computing” had not been popularized at this point, many of the principles I studied later became critical aspects of cloud computing and virtualization.

5. Since earning my doctorate degree over twenty-four years ago, I have been an employee, entrepreneur, and consultant in the computer systems, security, and cloud computing industry. I have worked extensively with technologies related to computer security, mobile security, malware detection, quantitative and qualitative fraud analysis, and disruptive security. I have worked as an engineer 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.

6. 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. Example publications of mine from this time period include “Secure distributed computation in cryptographic applications” (U.S. Patent No. 6,950,937, 2001 priority date) and “Secure server-aided signature generation” (International Workshop on Public Key Cryptography PKC 2001: Public Key Cryptography 383-401), both of which relate to virtualization. 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.

7. 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.

8. 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 man-in-the-middle attacks, online abuse in general, and countermeasures to attacks and abuses of these types.

9. 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.

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 authentication and cloud computing, 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. Using virtual machines is an important tool for security researchers, as it allows them to study the impact of a given executable, website, or other context on a computational environment. My students, direct reports and I have often relied on this approach in the context of interacting with online criminals, their websites and emails they send, as virtual machines enable us to identify their motives and report them to the authorities without making us easy for them to track or identify. Doing this is important to identify new and trending attacks.

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 as it pertains to web browsers. Relevant examples include:

- “Web Camouflage: Protecting Your Clients from Browser-Sniffing Attacks,” 5 IEEE Security & Privacy (Nov. 2007)
- “Invasive Browser Sniffing and Countermeasures,” Conference: 15th International Conference on World Wide Web (May 2006)
- “Remote Harm-Diagnostics”, (PDF) Privacy-preserving history mining for web browsers (researchgate.net)
- “Privacy-Preserving History Mining for Web Browsers”, (PDF) Privacy-preserving history mining for web browsers (researchgate.net)
- “Server-Side Detection of Malware Infection”, NSPW Proceedings on New Security Paradigms Workshop (2009), Server-side detection of malware

infection | Proceedings of the 2009 workshop on New Security Paradigms
Workshop (acm.org)

22. Related patent publications include U.S. Patent Nos. 10,498,735, 9,203,835, 8,578,174. These patents were assigned to PayPal, Inc. and 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 system monitoring, software security, attack detection, and network security.

II. LEGAL FRAMEWORK

25. I am a technical expert and do not offer any legal opinions. However, counsel has informed me as to certain legal principles regarding patentability and related matters under United States patent law, which I have applied in performing my analysis and arriving at my technical opinions in this matter.

26. I have been informed that claim terms are to be given the meaning they would have to a person having ordinary skill in the art at the time of the invention, taking into consideration the patent, its file history, and, secondarily, any applicable extrinsic evidence (e.g., dictionary definitions).

27. I have also been informed that the implicit or inherent disclosures of a prior art reference may anticipate the claimed invention. Specifically, if a person having ordinary skill in the art at the time of the invention would have known that the claimed subject matter is necessarily present in a prior art reference, then the prior art reference may “anticipate” the claim. Therefore, a claim is “anticipated” by the prior art if each and every limitation of the claim is found, either expressly or inherently, in a single item of prior art.

28. I have also been informed that a person cannot obtain a patent on an invention if the differences between the invention and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art. I have been informed that a conclusion of obviousness may be founded upon more than a single item of prior art. I have been further informed that obviousness is determined by evaluating the following factors: (1) the scope and content of the prior art, (2) the differences between the prior art and the claim at issue, (3) the level of ordinary skill in the pertinent art, and (4) secondary considerations of non-obviousness. In addition, the obviousness inquiry should not be done in hindsight. Instead, the obviousness inquiry should be done through the eyes of a person having ordinary skill in the relevant art at the time the patent was filed.

29. In considering whether certain prior art renders a particular patent claim obvious, counsel has informed me that I can consider the scope and content of the prior art, including the fact that one of skill in the art would regularly look to the disclosures in patents, trade publications, journal articles, industry standards, product literature and documentation, texts describing competitive technologies, requests for comment published by standard setting organizations, and materials from industry conferences, as examples. 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.

30. I have been informed that, in order to establish that a claimed invention was obvious based on a combination of prior art elements, a clear articulation of the reason(s) why a claimed invention would have been obvious must be provided. Specifically, I am informed that a combination of multiple items of prior art renders a patent claim obvious when there was an apparent reason for one of ordinary skill in the art, at the time of the invention, to combine the prior art, which can include, but is not limited to, any of the following rationales: (A) combining prior art methods according to known methods to yield predictable results; (B) substituting one known element for another to obtain predictable results; (C) using a known technique to improve a similar device in the same way; (D) applying a known technique to a known device ready for improvement to yield predictable results; (E) trying a finite number of identified, predictable potential solutions, with a reasonable expectation of success; (F) identifying that known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces if the variations are predictable to one of ordinary skill in the art; or (G) identifying an explicit teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify the prior art reference or to combine the prior art references to arrive at the claimed invention.

31. I am informed that the existence of an explicit teaching, suggestion, or motivation to combine known elements of the prior art is a sufficient, but not a

necessary, condition to a finding of obviousness. This so-called “teaching suggestion-motivation” test is not the exclusive test and is not to be applied rigidly in an obviousness analysis. In determining whether the subject matter of a patent claim is obvious, neither the particular motivation nor the avowed purpose of the patentee controls. Instead, the important consideration is the objective reach of the claim. In other words, if the claim extends to what is obvious, then the claim is invalid. I am further informed that the obviousness analysis often necessitates consideration of the interrelated teachings of multiple patents, the effects of demands known to the technological community or present in the marketplace, and the background knowledge possessed by a person having ordinary skill in the art. All of these issues may be considered to determine whether there was an apparent reason to combine the known elements in the fashion claimed by the patent.

32. I also am informed that in conducting an obviousness analysis, a precise teaching directed to the specific subject matter of the challenged claim need not be sought out because it is appropriate to take account of the inferences and creative steps that a person of ordinary skill in the art would employ. The prior art considered can be directed to any need or problem known in the field of endeavor at the time of invention and can provide a reason for combining the elements of the prior art in the manner claimed. In other words, the prior art need not be directed towards solving the same specific problem as the problem addressed by the patent. Further, the

individual prior art references themselves need not all be directed towards solving the same problem. I am informed that common sense is important and should be considered. Common sense teaches that familiar items may have obvious uses beyond their primary purposes.

33. I also am informed that the fact that a particular combination of prior art elements was “obvious to try” may indicate that the combination was obvious even if no one attempted the combination. If the combination was obvious to try (regardless of whether it was actually tried) or leads to anticipated success, then it is likely the result of ordinary skill and common sense rather than innovation. I am further informed that in many fields it may be that there is little discussion of obvious techniques or combinations, and it often may be the case that market demand, rather than scientific literature or knowledge, will drive the design of an invention. I am informed that an invention that is a combination of prior art must do more than yield predictable results to be non-obvious.

34. I am informed that for a patent claim to be obvious, the claim must be obvious to a person of ordinary skill in the art at the time of the invention. I am informed that the factors to consider in determining the level of ordinary skill in the art include (1) the educational level and experience of people working in the field at the time the invention was made, (2) the types of problems faced in the art and the

solutions found to those problems, and (3) the sophistication of the technology in the field.

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 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; (h) teaching away from the invention in the prior art.

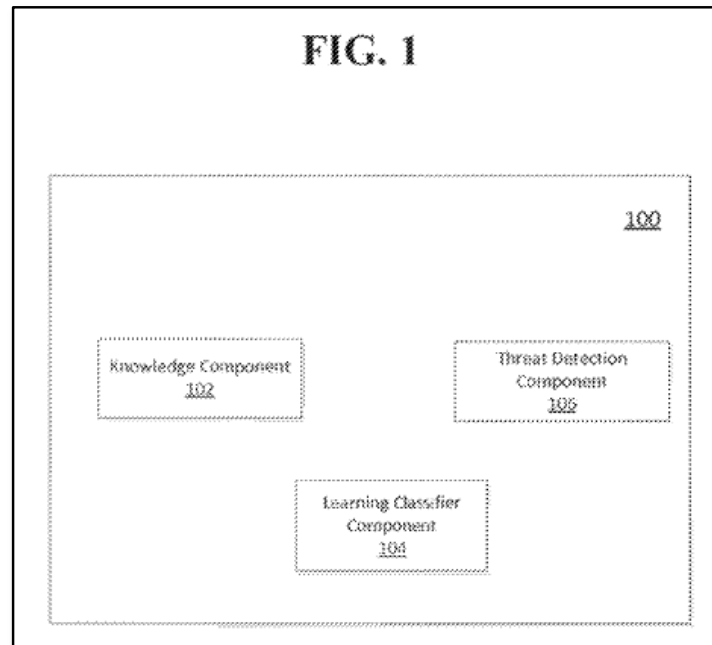
37. I am further informed that secondary considerations evidence is only relevant if the offering party establishes a connection, or nexus, between the evidence and the claimed invention. The nexus cannot be based on prior art features. The establishment of a nexus is a question of fact. While I understand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

III. OPINION

A. Overview of the '869 Patent

38. The '869 Patent relates generally to threat detection of executable files. '869 Patent (Ex. 1001), 1:31-45, 2:23-36. Specifically, the '869 Patent describes an analysis of an executable file that involves extracting static data points, such as file size, file name, file version, etc., and using a classifier to analyze this extracted static

data. For example, the '869 Patent explains that its threat detection analysis analyzes threats by “evaluating static data points extracted from an executable file” and uses a “trained classification system” to analyze this data. *Id.*, 2:36-53. The '869 Patent provides an exemplary system, as shown below, for accomplishing these analyses:



Id., Fig. 1.

39. The '869 Patent describes several exemplary embodiments, one including where this system analyzes a portable executable (PE) file for threats. *Id.*, 3:46-54. I note that this system is not limited to PE files, such that the '869 Patent provides that its exemplary system can analyze “any file or program that executes a task according to an encoded instruction.” *Id.* To analyze threats, knowledge component 102 “collect[s] data for use in building, training and/or re-training a

learning classifier to evaluate executable files.” *Id.*, 3:54-60, 4:29-44. This includes collecting any number of static data points. *Id.*, 5:62-6:44. Some example static data points are shown below in Table 1.1:

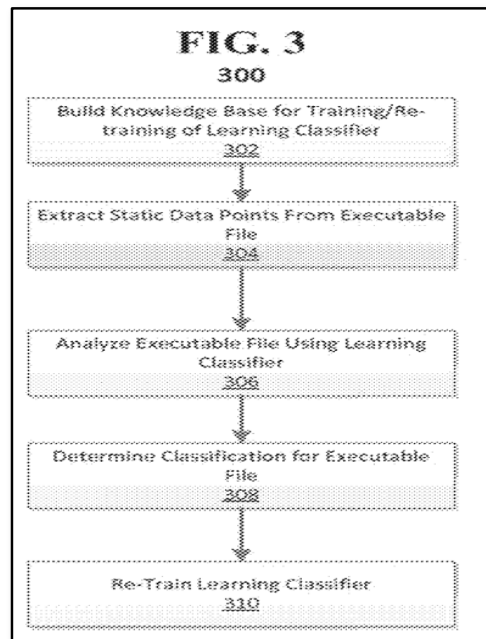
TABLE 1.1			
Numeric values	Nominal values	Strings/Byte sequences	Boolean values
File size	initialize	Comments	Address Of Entry Point Anomaly
linker version	Un-initialize	company name	Image Base Anomaly
code size	entry point	file description	Section Alignment Anomaly
OS version	subsystem	internal name	Size Of Code Mismatch Anomaly
image version	file subtype	legal copyright	Low Import Count Anomaly
subsystem version	language	original file	Entry Point Anomaly
file version number	file flags masks	private build	certificate Validity
product version number	file flags	product name	Certificate Exception
size of heap	file OS	special build	Code Characteristics Anomaly
size of stack	file type	product version	Code Name Anomaly
size of image	machine type	file version	Count Anomaly
PE header time	PE type	package code	Data Characteristics Anomaly
Section Entropy	section counts	product code	Data Name Anomaly
Sections count	DLL count	export DLL name	Export Exception
	DLL functions	assembly version	Large Number of DLLs Anomaly
	data directory	Certificate Issuer	flag DLL Name Anomaly
	export count	Certificate Subject	Number of Functions Anomaly
	Earliest Data Byte	Imports	Function Name Anomaly
	resources	Exports	PE Header Anomaly
	resources language	Section Names	High Section Count Anomaly
	resource Encoding	Non-resource section strings	PE Magic Validity
	resource code page		Resource Exception
	resource size		VR Code Ratio Anomaly
	DLL characteristics		Import Exception

Id., Table 1.1.

40. Then, learning classifier component 104, which is trained with known threat, benign, and unwanted data, evaluates the executable file “using the information provided by the knowledge component 102” through “programming operations or machine learning processing.” *Id.*, 7:26-34, 8:4-7. The learning classifier can also “adaptively set features...based on the static data points” and “controllably select (e.g., turn on/off) features for evaluation.” *Id.*, 7:43-48. For example, “[i]n one instance where a static data point extracted for evaluation is a file

size (e.g., numeric value), the learning classifier may turn on/off features for evaluation based on whether [] the file size of the file is within a certain range.” *Id.*, 7:48-52.

41. Additionally, the ’869 Patent explains that to make the static data points suitable for the above processing operations, learning classifier 104 processes this data. *Id.*, 8:8-10. As part of this processing, “string sequence data as well as byte sequence data may be parsed and processed as n-grams” such that “the unique n-grams string and/or byte sequences are then used as input to a generated feature vector.” *Id.*, 8:16-23. I note that in the Challenged Claims, this feature vector is “input to a support vector machine (SVM)” or a “machine learning model” for categorization. *Id.*, 9:27-35. Finally, threat detection component 106 evaluates the results of the learning classifier and makes a “determination as to whether the executable file is malicious, benign or a potentially unwanted file/application.” *Id.*, 9:55-59. An exemplary threat detection method is shown below in Figure 3:



Id., Fig. 3.

B. Overview of the Relevant File Histories

42. I understand that the application which issued as the '869 Patent—Application No. 16/791,649 (the “'649 Appl.”)—was filed on February 14, 2020, and claims priority to an application filed on May 12, 2015. I apply this May 12, 2015, date for the purposes of my opinions herein only as the priority date for the Challenged Claims.

43. I understand the '649 Appl. which issued as the '869 Patent is a “child” continuation of U.S. Patent No. 10,599,844 (referred to herein as the “parent”). I have been informed that this means the '869 Patent shares the same specification as the '844 Patent.

1. Overview of the '844 Parent Patent's File History

44. The claims of the '844 Patent were rejected several times over the prior art, including several rejections involving US 8,709,924 to Wojnowicz et al. ("Wojnowicz"). '844 FH (Ex. 1006), 92-108, 147-166, 209-229, 265-284. Per Applicant, Wojnowicz:

relates to identifying malware using wavelet decomposition of software entropy. Specifically, Wojnowicz teaches receiving a set of data files. Each of the data files is split into non-overlapping file chunks of fixed length. The non-overlapping file chunks are represented as an entropy time series that reflects an amount of entropy across the file chunks for each file. A wavelet transform is applied to the corresponding entropy time series to generate an energy spectrum characterizing, for the file, an amount of entropic energy at multiple scales of code resolution. A determination is then made as to whether the file is likely to be malicious based on the energy spectrum. Wojnowicz, col. 1, 11. 23-31.

Id., 134. Applicant first amended its claims to "clarify that the static data points extracted from the received executable file correspond to points of reference (in the file content) that are used to evaluate the file" such that Wojnowicz's file chunks split into "a fixed length boundary (e.g., 256 bytes)" "do not correspond to particular sections or portions of the file." *Id.*, 135. The Examiner rejected Applicant's arguments and issued a final rejection, explaining an example static data point is "string information" and that "Wojnowicz computes entropy of a string of hexadecimal characters." *Id.*, 147-149 (further noting "the location of the string within the executable file is static").

45. Applicant filed an RCE and further amended the claims to state that “the one or more points of reference are defined by the executable file.” *Id.*, 185-186. Applicant argued that Wojnowicz’s fixed length boundaries for its chunks “occur randomly” and cannot be defined by the executable file. *Id.* The Examiner again rejected these arguments, noting “Wojnowicz splits the file into sections 256 byte apart, thus a point of reference is defined at each portion of the file located at each consecutive 256 byte location. Expressed differently, this point of reference is defined by the executable file because it is based upon the executable file and the number of sequential bytes counted in the executable file to define the each point.” *Id.*, 211. Applicant thereafter made a similar amendment, requiring “static data points are points of reference *representing sections of the executable file*,” suggesting because Wojnowicz’s chunks “may comprise data from different sections of the file,” these chunks do not represent sections of the file, which was similarly rejected by the Examiner. *Id.*, 249, 265-268.

46. Applicant filed another RCE, removing the “points of reference” language and adding “*the plurality of static data points represent predefined character strings in the executable file*.” *Id.*, 305. Applicant argued that Wojnowicz’s “fixed-length file chunks do not represent predefined character strings” where “each fixed-length file chunk is, itself, a random (e.g. undefined) character string having no initial significance to the system of Wojnowicz.” *Id.*, 312.

47. The Examiner rejected the amended claims with new references, and Applicant presented further amendments, adding the “generating a feature vector...” limitation found in the issued claims including the language requiring a feature of the feature vector to be “selectively turned on or off based on whether a value of one or more static data points...is within a predetermined range.” *Id.*, 372. I understand that this positioned the claims for allowance. *Id.*, 387-395.

2. Overview of the '869 Patent's File History

48. I understand that the independent claims of the '869 Patent omit several of the concepts recited in the independent claims of the parent '844 Patent, such as requiring “static data points” to be “predefined character strings” or limiting extraction of static data points to be without unpacking or decrypting. In contrast, the challenged independent claims were amended to include the following limitation (also found in the parent '844 Patent claims, as noted above) to traverse the cited art of record: “wherein one or more features of the feature vector are selectively turned on or off *based at least in part on evaluation of whether a value of one of the plurality of static data points is within a predetermined range.*” '869 FH (Ex. 1002), 217, 226. This limitation purportedly differentiated the claimed invention from both “prior art discussing simply disregarding/ignoring features of a data set as a means to avoid skewing classification” (*see id.*, 121 discussing US2011/0172504 to Wegerich) and “selectively turning on/off one or more features...based on the number of classifiers

associated with the features for the benefit of efficiency” (*see id.*, 259 discussing US2010/0082642 to Foreman).

49. I understand a Notice of Allowance issued thereafter, citing the above limitation as the reason for allowance. *Id.*, 253-259.

C. Level of a Person Having Ordinary Skill in the Art

50. I have been asked to provide my opinion as to the level of skill of a person having ordinary skill in the art at the time of the '869 Patent, which I have been instructed to assume is May 12, 2015. In determining the characteristics of a hypothetical person having ordinary skill in the art at the time of the claimed invention, I was told to consider several factors, including the type of problems encountered in the art, the solutions to those problems, the rapidity with which innovations are made in the field, the sophistication of the technology, and the education level of active workers in the field. I also placed myself back in the time frame of the claimed invention and considered the colleagues with whom I had worked at that time.

51. In my opinion, a person having ordinary skill in the art, as of May 12, 2015, would have been a person having a bachelor's degree in computer science, computer engineering, or an equivalent, as well as two years of industry experience and would have had a working knowledge of host monitoring systems, software security analysis, and dynamic malware analysis. Additional graduate education

could substitute for professional experience, or significant experience in the field could substitute for formal education. Such a person would have been capable of understanding the '869 Patent and the prior art references discussed below.

52. Based on my education, training, and professional experience in the field of the claimed invention, I am familiar with the level and abilities of a person of ordinary skill in the art at the time of the claimed invention. Additionally, I met at least these minimum qualifications to be a person having ordinary skill in the art as of the time of the claimed invention of the '869 Patent.

D. Claim Construction

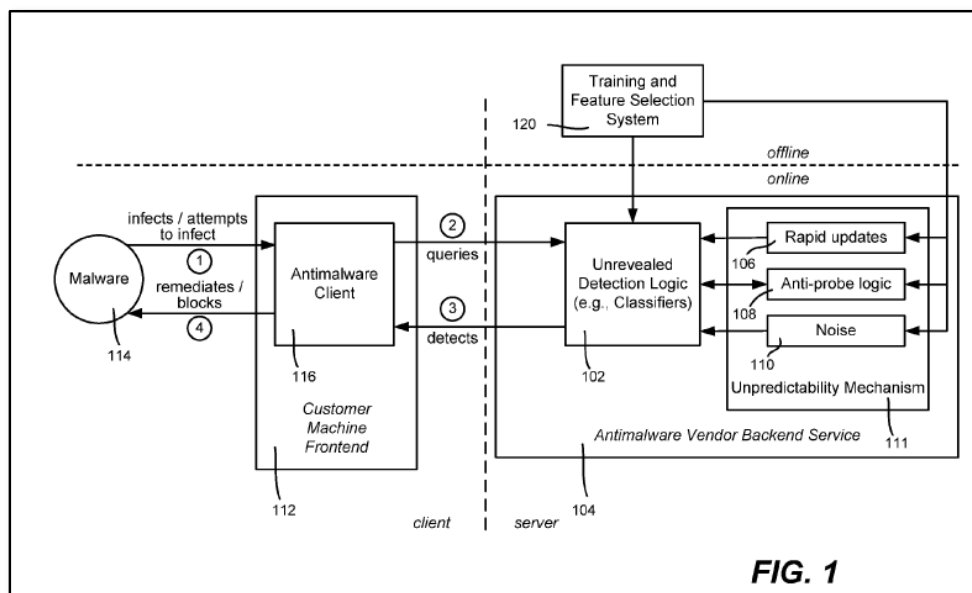
53. I have been informed by counsel that the first step in a patentability analysis involves construing the claims, as necessary, to determine their scope. Second, the construed claim language is then compared to the disclosures of the prior art. I am informed that claims are generally given their ordinary and customary meaning as understood by a person having ordinary skill in the art at the time of the invention, in light of the patent specification.

54. I understand that in parallel litigation, Patent Owner contends that no claim language requires express construction. For the purposes of my analysis, I have been instructed to apply the plain and ordinary meaning to the claim terms at issue herein unless noted otherwise.

E. Summary of the Prior Art References

1. Kumar

55. *Kumar* (Ex. 1004) discloses threat detection systems and methods which utilize “antimalware detection logic [] maintained and operated at a backend service...for purposes of malware detection.” *Kumar*, Abstract, [0005], [0018]. For example, an exemplary system is depicted below:



Id., Fig. 1.

56. *Kumar* discloses that its system performs a method “at least in part on at least one processor,” comprising:

receiving a malware-related query at a backend service, processing data associated with the malware-related query at the backend service, including via updateable detection logic that is not revealed outside of the backend service, to determine whether the data associated with the malware-related query corresponds to detected malware, and returning a corresponding result in response to the

query indicating whether the data associated with the malware-related query corresponds to detected malware.”

Kumar, Claim 1.

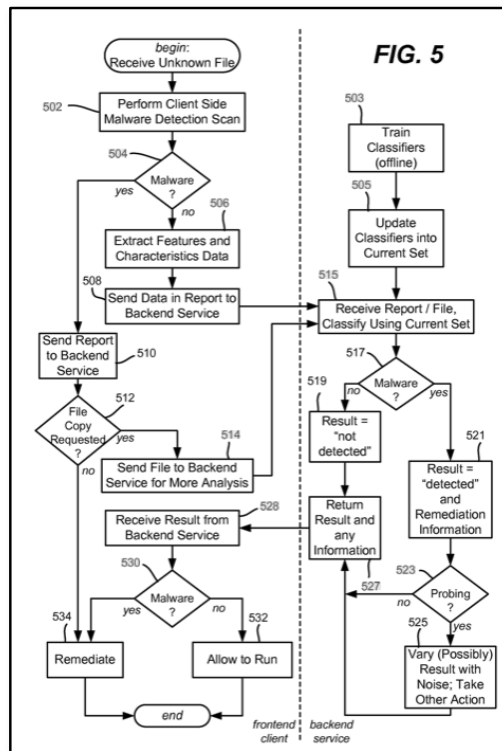
57. For example, in operation, anti-malware client component 116 on the frontend machine 112 receives an unknown file and performs processing before the file is allowed to run. *Id.*, [0022]-[0023], Fig. 5. In embodiments, component 116 extracts and sends information comprising characteristics and features of potential malware executables to the backend service in the form of a “telemetry report.” An exemplary telemetry report is provided below, though *Kumar* contemplates any number of static data points can be extracted from a file:

Feature	Description
File Name	Name of the PE file
Original File Name	File name in the original report
File Name Matches Original?	Whether the file name in this report matches the file name in the original report
File Type	Type of file it is
Signer Name	Organization that signed the file
Signing Authority	Certificate authority that issued the signature
Signature Type	File signed or not; if signed, whether the signature is legitimate or invalid
Description	Description of the file in the header
Organization	Manufacturer of the binary file
Version	Version number of the binary file
Packed	Whether the file is packed
LS Hash	Locality sensitive hash
Behavior Feature	Represents the simple behavior that caused the report to be generated

Id., [0026]-[0028] (noting these are “example features”), [0035], [0055] (“the features described herein are only examples”).

58. Then, the backend service 104 receives this data and processes it using detection logic 102. *Id.*, [0026]. Detection logic 102 “includes one or more trained telemetry report classifiers that determine, in part, whether the features and characteristics are indicative of malware.” *Id.*; *see also id.*, [0027] (listing exemplary classifiers, including a “support vector machine”). To accomplish this, feature data is preprocessed into a feature vector, the classifier analyzes this feature vector based on training data (*see* Fig. 2 for training process), and outputs a result, which may be “a probability” that the file “belong to a particular malware family, the generic malware family, a particular benign family, or the generic benign family.” *Id.*, [0036]-[0045], [0052], [0068]-[0071], Fig. 5.

59. In Figure 5 below, *Kumar* displays example steps for detecting malware:



Id., Fig. 5.

60. A PHOSITA would have recognized that each of *Kumar* and the '869 are related to systems and methods for threat detection of executable files, and thus are in the same field of endeavor. *Kumar*, Abstract, [0005], [0018], [0036]-[0045], [0052], [0068]-[0071], Fig. 5; '869 Patent 1:31-45, 2:23-53, 3:46-60, 4:29-44, 5:61-67, 7:26-34, 8:4-7, 16:10-33. A PHOSITA also would have recognized *Kumar* is directed at solving similar problems to those of the '869 Patent, including detecting threats without decrypting and unpacking unknown files and/or using static data points to analyze a file. *Compare Kumar*, [0039], [0054] (discussing analysis

without unpacking), [0028] (identifying exemplary static data points for analysis),
with '869 Patent, 1:31-36, 2:23-53, 12:11-45, claims 2, 11, 16, 19, 33, 47.

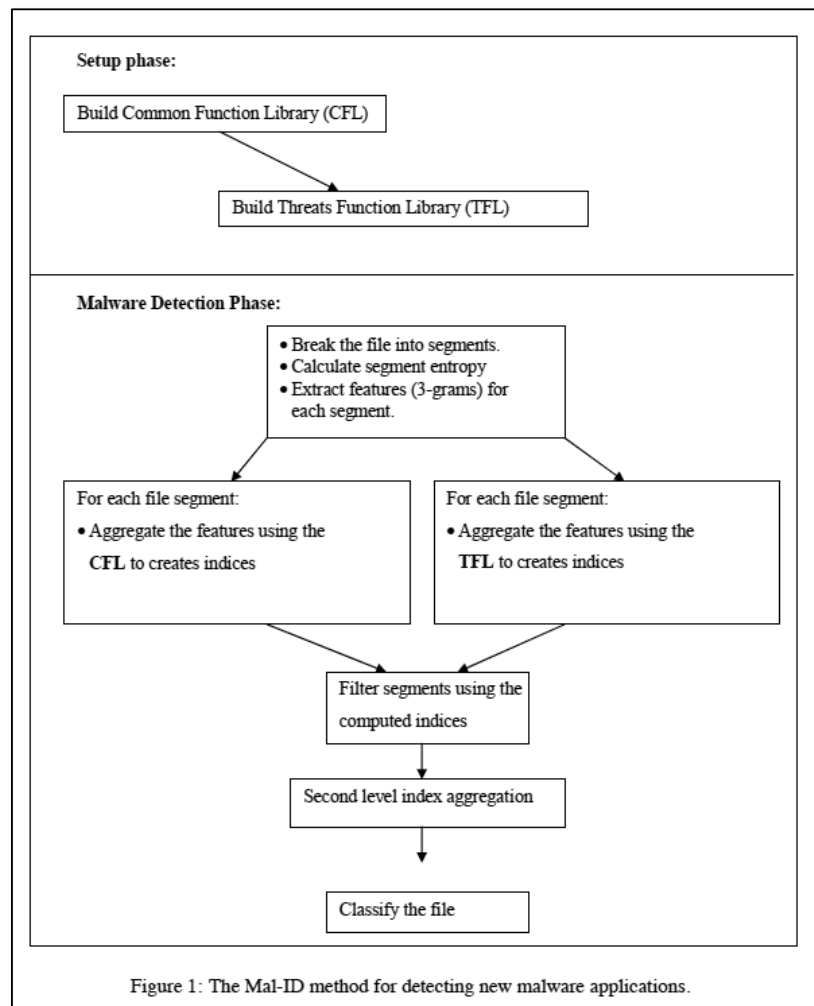
2. Tahan

61. I have been informed that *Tahan* (Ex. 1005) was publicly available by at least April 2012. I understand that Petitioner provides the declaration of Ms. June Munford and Dr. Ingrid Hsieg Yee in support. *Munford Dec.* (Ex. 1008); *Dr. Yee Dec.* (Ex. 1015). I have reviewed the opinions of Ms. Munford and Dr. Yee, which explain that *Tahan* was published and made available in the *Journal of Machine Learning Research*, 2012. I note that this journal was well-known in the computer security and threat detection industry and was widely available and accessible to those of skill in the art and other interested persons, such that PHOSITA could have accessed this journal if interested in the subject matter disclosed in that journal.

62. *Tahan* is directed to a threat detection system and method which detects malicious files without unpacking the file. *Tahan*, Abstract, 949 (describing “methods, based on machine learning, to detect malware in executable files without any need for preprocessing, such as unpacking or disassembling”). In embodiments, *Tahan*’s system/method utilizes a “static (form-based) analysis methodology” termed “Mal-ID” which uses common segment analysis...to detect malware files.” *Id.*, 949. “The Mal-ID basic is a feature extraction process followed by a simple static decision rule. It operates by analyzing short segments extracted from the file

examined. Each segment comprises a number of 3-grams depending on the length of the segment (e.g., a segment of length 4 bytes is comprised from two 3-grams that overlap by two bytes). Three features can be derived for each segment: Spread, MFG, and Entropy.” *Tahan*, 957.

63. Mal-ID’s methodology is separated into two phases: (1) a “Setup Phase” and (2) a “Malware Detection Phase.” *Id.*, 955-956. This is shown below in Figure 1:



Id., 956.

64. During Setup, *Tahan* builds the repositories requisite for the Detection phase, including a “Common Function Library” (CFL) which contains benign files and a “Threat Function Library” (TFL) which contains malware files. *Id.*, 955. Then, during Detection, a file is broken into 64-byte segments, these segments are broken into 3grams (three consecutive bytes), and features are derived for each segment including “Spread, MFG, and Entropy.” *Id.*, 955-959. These features are “meta-features” “used to represent features of features (features of the basic 3-grams).” *Id.*, 957. The system then uses a detection algorithm to a classify a file as benign or malware. *Id.*, 958-958; *see also id.*, 960 (providing pseudocode for this algorithm). During evaluation, this algorithm removes irrelevant segments by setting thresholds for segment entropy values, including EntropyLow and EntropyHigh such that entropy’s falling below the low threshold or above the high threshold are discarded. *Id.*, 958, 960.

65. A PHOSITA would have recognized that *Tahan* and the ’869 Patent are related to systems and methods for threat detection of executable files, and thus are in the same field of endeavor. *Compare Tahan*, 949, 955-959, Figures 1 and 2 *with* ’869 Patent, 1:31-45, 2:23-53, 3:46-60, 4:29-44, 7:26-34, 8:4-7. A PHOSITA also would have recognized *Tahan* is directed at solving problems similar to those of the ’869 Patent, such as detecting threats without decrypting and unpacking unknown

files. *Compare Tahan*, 949, 975 (discussing analysis without unpacking), [0028] (identifying exemplary static data points for analysis), *with* '869 Patent, 1:31-36, 2:23-53, 12:11-45, claims 2, 11, 16, 19, 33, 47.

F. Opinions Regarding Ground 1: Claims 1-59 are rendered obvious over *Kumar* or, Alternatively, *Kumar* in view of *Tahan*.

1. *Claim 1*

(a) *1[Pre]. A computer-implemented method comprising:*

66. *Kumar* describes a computer-implemented method which involves “determin[ing] whether the data associated with the malware-related query corresponds to detected malware[.]” *Kumar*, [0005]; *see also id.*, Claim 1 (“In a computing environment, a method performed at least in part on at least one processor...”).

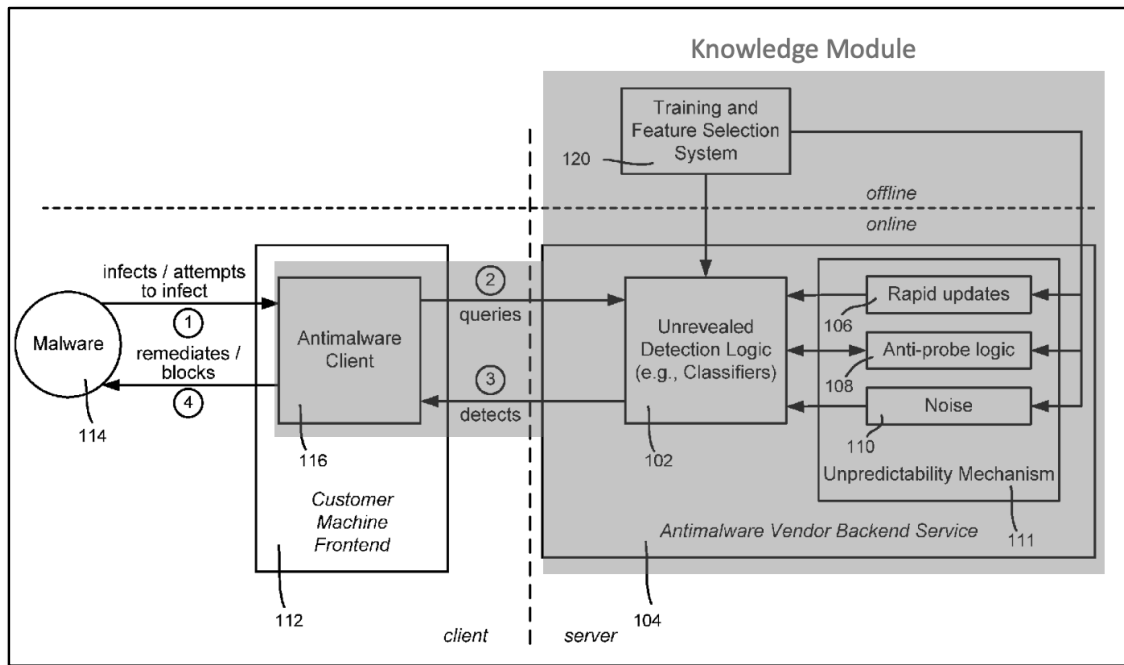
(b) *[1(a)] identifying, by a knowledge module, static data points that may be indicative of either a harmful or benign executable file;*

67. *Kumar* describes an antimalware detection system that identifies static data points that are indicative of either a harmful or benign executable file.

68. In this antimalware detection system, *Kumar* utilizes a computing and networking environment. *Kumar* explains that computer-readable media may comprise computer storage media and communication media. *Kumar*, [0077]. As discussed above, *Kumar*’s antimalware detection system comprises one or more storages and memories.

69. I note that the '869 Patent does not define a “knowledge module,” but it generally explains that a “knowledge component 102 is one or more storages, memories, and/or modules that continuously collects and manages data that may be used to detect threats in files such as executable files.” '869 Patent, 1:59-61. I note that the '869 Patent's components “may be spread across multiple devices. For instance, files to be evaluated may be present on a client device and information may be processed or accessed from other devices in a network such as, for example, one or more server devices that may be used to perform threat detection processing and/or evaluation a file before execution of the file by the client device.” '869 Patent, 3:24-30.

70. Consistent with the '869 Patent's disclosures, a PHOSITA would have understood that *Kumar's* antimalware detection system, which is made up of multiple modules for collecting and managing data to detect threats in files, is a “knowledge module,” as described in the '869 Patent. *Kumar's* knowledge module includes an antimalware client 116 running on a customer's local machine and an antimalware backend service 104 running on a server:



Id., Fig. 1 (annotated).

71. *Kumar* discloses that anti-malware client component 116 on the frontend machine 112 receives an unknown file and performs processing before the file is allowed to run. *Id.*, [0022]-[0023], Fig. 5. In embodiments, component 116 extracts and sends information comprising characteristics and features of potential malware executables to the backend service. *Id.*, [0026]-0028]. The backend service 104 receives this data and processes it using detection logic 102. *Id.*, [0026]. Detection logic 102 “includes one or more trained telemetry report classifiers that determine, in part, whether the features and characteristics are indicative of malware.” *Id.* Thus, *Kumar*’s antimalware detection system comprises modules that

continuously collect and manage data that may be used to detect threats in files such as executable files as described by the '869 Patent.

72. Additionally, knowledge module identifies several sets of static data points to train a classifier used to predict whether an unknown file is malware or benign. For example, in one implementation, “[r]aw telemetry logs 222 (reports) and analyst labels 224 are input to [training and feature selection] system 120.” *Kumar*, [0051]. “Some example features derived from the telemetry data and used to train the classifiers” are shown below:

Feature	Description
File Name	Name of the PE file
Original File Name	File name in the original report
File Name Matches Original?	Whether the file name in this report matches the file name in the original report
File Type	Type of file it is
Signer Name	Organization that signed the file
Signing Authority	Certificate authority that issued the signature
Signature Type	File signed or not; if signed, whether the signature is legitimate or invalid
Description	Description of the file in the header
Organization	Manufacturer of the binary file
Version	Version number of the binary file
Packed	Whether the file is packed
LS Hash	Locality sensitive hash
Behavior Feature	Represents the simple behavior that caused the report to be generated

Id., [0028]. Moreover, an additional “Boolean feature” can be used “to indicate if a particular attribute is blank or null.” *Id.* Additionally, “[s]ome measure of the

differences in the file names such as the entropy of the observed filenames also may be used as a feature.” *Id.*, [0035].

73. The features in *Kumar*, “can be used to classify if an individual report corresponds to a malicious file.” *Kumar*, [0035].

74. In other examples, *Kumar*’s system “uses system state variables, unpacked strings, API system call trigrams, and API system calls plus an individual parameter value as potential features[.]” *Id.*, [0055]. *Kumar* notes “that the features described herein are only examples, and the classifier is not limited to these types of features, as others (e.g., file geometry, file reputation and so forth) may be used instead of or in addition to some or all of the ones exemplified herein.” *Id.*

75. In each example, training and feature selection system 120 “includes various processing blocks including a feature selection component 226” that “determines a subset of the total features to be used for classification.” *Id.*, [0051], [0055]. Here, for example, “a feature selection algorithm attempts to choose the 3,000 ‘best’ features per family, and 12,000 ‘best’ features for the generic malware and benign families (where ‘best’ may be determined according to an appropriate ranking metric).” *Id.* Thus, *Kumar*’s selected features may be indicative of either a malicious (i.e., *harmful*) or benign executable file. Indeed, once these features are identified and collected, a “labeled dataset is constructed from the analysts’ labels 224” and the feature data and is “used to train a classifier model 232[.]” *Id.*, [0052]

(describing using telemetry feature data for training); [0055] (describing using file execution feature data for training). To train the model, a preprocessing step converts the raw data to the feature vector used to train the classifier. *Id.*, [0052], [0055]. The output of the training system comprises a classifier model “(e.g., a set of weights or parameters), which can be used to **predict if an unknown file belongs to a particular malware family, a generic malware class, a particular benign family, or a generic benign class.**” *Id.*; *see also id.*, [0068], Fig. 5 (steps 503 and 505).

76. I note that the '869 Patent discloses “static data points include, but are not limited to header information, section information, import and export information, certificate information, resource information, string and flag information, legal information, comments, and/or program information (e.g., APIs and DLLs), among other examples.” '869 Patent, 5:61-67. Further, the '869 Patent provides Table 1.1 which “identifies some of the static data points identified for analysis of an executable file[.]” '869 Patent, 6:15-18. Table 1.1 is below:

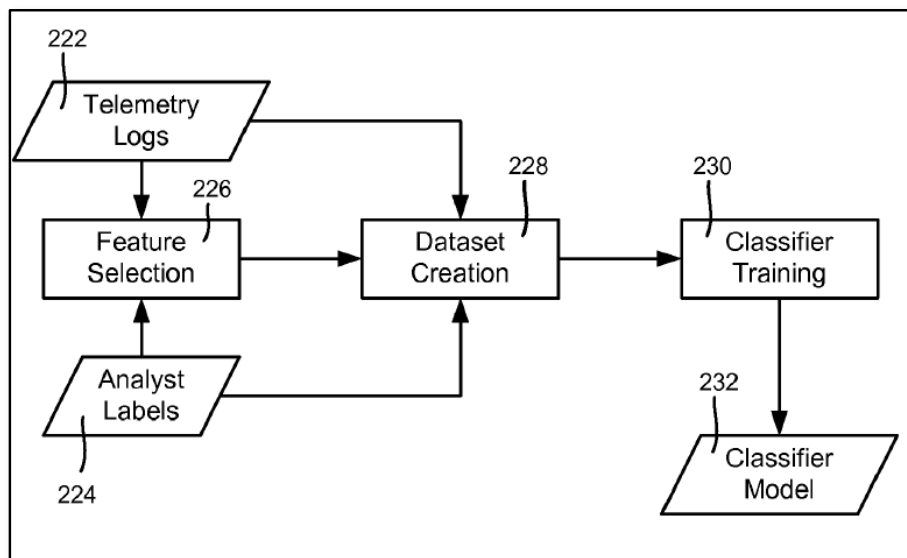
TABLE 1.1			
Numeric values	Nominal values	Strings/Byte sequences	Boolean values
File size	initialize	Comments	Address Of Entry Point Anomaly
linker version	Un-initialize	company name	Image Base Anomaly
code size	entry point	file description	Section Alignment Anomaly
OS version	subsystem	internal name	Size Of Code Mismatch Anomaly
image version	file subtype	legal copyright	Low Import Count Anomaly
subsystem version	language	original file	Entry Point Anomaly
file version number	file flags masks	private build	certificate Validity
product version number	file flags	product name	Certificate Exception
size of heap	file OS	special build	Code Characteristics Anomaly
size of stack	file type	product version	Code Name Anomaly
size of image	machine type	file version	Count Anomaly
PE header time	PE type	package code	Data Characteristics Anomaly
Section Entropy	section counts	product code	Data Name Anomaly
Sections count	DLL count	export DLL name	Export Exception
	DLL functions	assembly version	Large Number of DLLs Anomaly
	data directory	Certificate Issuer	flag DLL Name Anomaly
	export count	Certificate Subject	Number of Functions Anomaly
	Earliest Data Byte	Imports	Function Name Anomaly
	resources	Exports	PE Header Anomaly
	resources language	Section Names	High Section Count Anomaly
	resource Encoding	Non-resource section strings	PE Magic Validity
	resource code page		Resource Exception
	resource size		VR Code Ratio Anomaly
	DLL characteristics		Import Exception

'869 Patent, Table 1.1. Many of *Kumar's* features, including file names, file description, file type, version information, Boolean values certificate information, entropy, etc. are identified as *static data points* by the '869 Patent. A PHOSITA would have understood *Kumar's* identified features to be *static data points* within the definition provided by the '869 Patent. Moreover, *Kumar's* identified features also fall within the description of static data points during the '844 parent patent's prosecution. As I stated above in my summary of the '844 parent file history, applicant explained that "static data points extracted from the received executable file correspond to points of reference (in the file content) that are used to evaluate the file." Consistent with this, *Kumar's* static data points correspond to points of reference in the file content, such as file header information, for example.

- (c) [1(b)] associating, by the knowledge module, the identified static data points with one of a plurality of categories of files, the plurality of categories of files including harmful files and benign files;

77. As I described above in claim 1(a), during *Kumar*'s training process, several features (i.e., static data points) are identified and collected, and training system 120 constructs a labeled dataset from these features and the analyst labels to train the classifier. I note that the training system 120 is a component of the "knowledge module."

78. For example, *Kumar*'s training process is shown below:



Id., Fig. 2.

79. Here, *Kumar* associates the identified static data points with categories of files, such as harmful and benign files. Specifically, *Kumar*'s training system receives raw telemetry logs and analyst labels as input. *Kumar*, [0051]. The training

system includes various components for processing this input, including a feature selection component, a dataset creation component, and a classifier training component. *Kumar*, [0051]. The feature selection component in *Kumar* determines “a subset of the total features to be used for classification.” *Kumar*, [0051]. For instance, “[i]n one example implementation, a feature selection algorithm attempts to choose the 3,000 “best” features per family, and 12,000 “best” features for the generic malware and benign families (where “best” may be determined according to an appropriate ranking metric).” *Kumar*, [0055]. The feature selection process is the same for the telemetry and file execution implementations. As an example, *Kumar* describes:

Thus, a ranked list of potential features for each family is computed, and used to choose the top K (3000, 12000) features for the family. **These top K features are a mixture of different types of features, depending on how well they discriminate the one family from other malware and benign file classes (e.g., other malware families, generic malware family, other generic families, and generic benign family).**

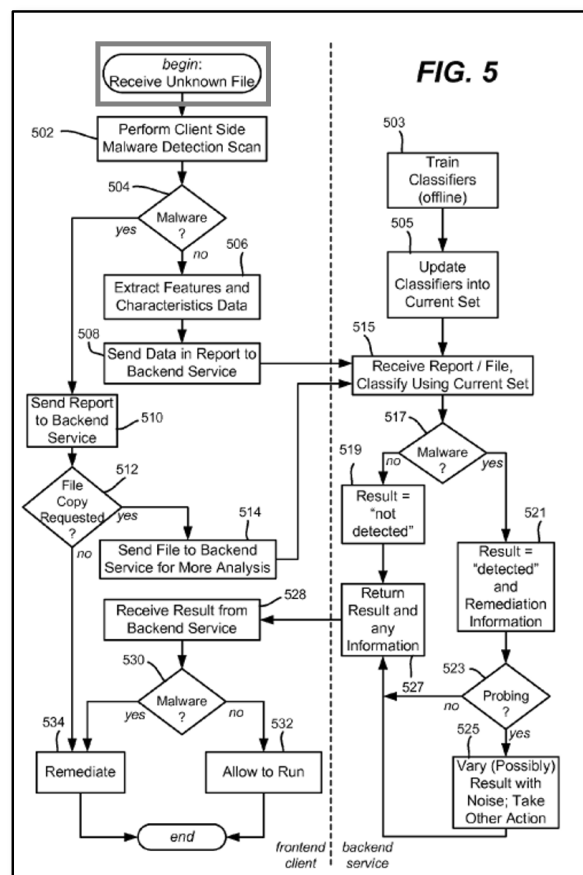
Kumar, [0056]. Then, “[t]he output of the training system comprises a classifier model 232 (e.g., a set of weights or parameters), which can be used to predict if an unknown report was generated by a particular malware family, a generic malware class, a particular benign family, or a generic benign class.” *Kumar*, [0052].

80. A PHOSITA would have understood that *Kumar*’s knowledge module (which includes the classification training system) therefore associates the identified

static data points with one of a plurality of categories of files, the plurality of categories of files including harmful files (i.e., malware) and benign files.

(d) [1(c)] identifying an executable file to be evaluated;

81. *Kumar* discloses that “[w]hen potential malware is received at the customer machine front-end 112, the potential malware is processed before being allowed to run, so as to detect whether the potential malware is actual malware 114.” *Kumar*, [0022]. As shown in Figure 5, the frontend client identifies an unknown executable file for evaluation after it is received at the front-end machine:



Id., Fig. 5; *see generally, id.*, [0026] (describing sending features for “executables” to a backend service), [0029] (backend service receiving a “report of the executable”), [0032] (transmitting telemetry reports for received “[e]xecutable files” to backend service).

- (e) *[1(d)] extracting, by the knowledge module, a plurality of static data points from the identified executable file;*

82. As I described in my opinions above in limitation 1(a), *Kumar*’s knowledge module includes a frontend client 116. *Kumar* discloses that “[w]hen potential malware is received at the customer machine front-end 112, the potential malware is processed before being allowed to run, so as to detect whether the potential malware is actual malware 114.” *Kumar*, [0022]. *Kumar* describes that “[a] frontend client may extract the features and send feature data with the malware-related query to the backend service.” *Kumar*, [0006]. To accomplish its methods, *Kumar* describes:

“More particularly, features and characteristics may be extracted from the file data by the antimalware client component 116, sent to the backend service 104, (e.g., in an xml report), and processed at the backend malware detection logic 102. To this end, the backend malware detection logic 102 includes one or more trained telemetry report classifiers that determine, in part, whether the features and characteristics are indicative of malware, or more specifically (in one implementation) a particular family of malware.”

Kumar, [0026]. In some embodiments, this feature extraction step in *Kumar* is “performed at the client to extract the features and any other characteristics data from the file” *Kumar*, [0067]; *see also id.*, Fig. 5 (step 506), claim 4.

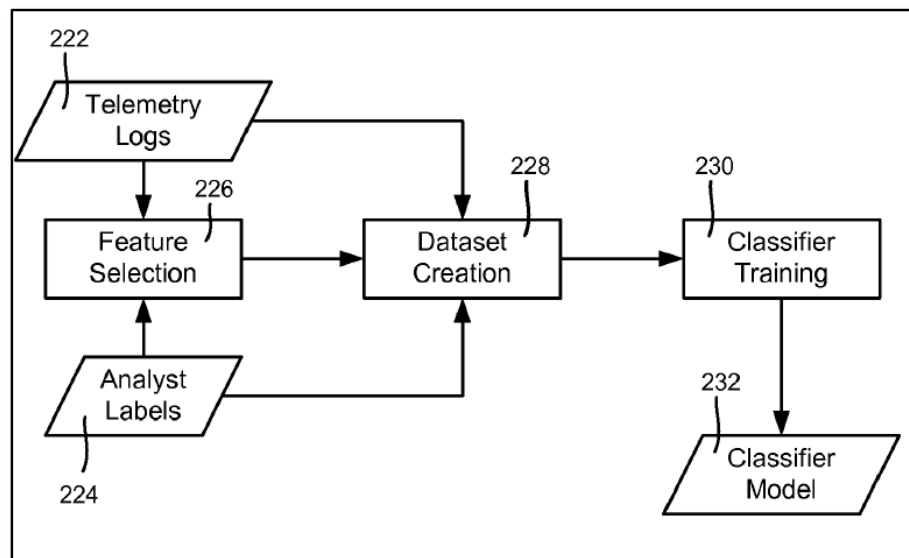
83. As I described above in my opinions in limitation 1(a), this feature data includes static data points, such as file names, file description, file type, version information, certificate information, Boolean values, entropy, etc. This data is shown in the table below:

Feature	Description
File Name	Name of the PE file
Original File Name	File name in the original report
File Name Matches Original?	Whether the file name in this report matches the file name in the original report
File Type	Type of file it is
Signer Name	Organization that signed the file
Signing Authority	Certificate authority that issued the signature
Signature Type	File signed or not; if signed, whether the signature is legitimate or invalid
Description	Description of the file in the header
Organization	Manufacturer of the binary file
Version	Version number of the binary file
Packed	Whether the file is packed
LS Hash	Locality sensitive hash
Behavior Feature	Represents the simple behavior that caused the report to be generated

Kumar, [0028]. And as I noted above in limitation 1(a), this feature data is consistent with the features described in the '869 Patent's specification and its parent '844 patent file history.

- (f) [1(e)] generating a feature vector from the plurality of static data points using a classifier trained to classify the static data points based on training data, the training data comprising files known to fit into one of the plurality of categories of files,

84. As noted in my opinions above, *Kumar* discloses a training process where a classifier is trained with training data, which includes telemetry logs and analyst labels. *See above*, Claim 1(a)-1(b). In particular, “a labeled dataset is constructed from the analysts’ labels 224 and the telemetry logs 222, and used to train a classifier model 232 using various algorithms.” *Kumar*, [0052]. Figure 2 depicting this process is reproduced below:



Id., Fig. 2.

85. I note that *Kumar*’s training data used to train its classifiers comprises files known to fit into different families of malware and benign files (i.e., *files known to fit in one of the plurality of categories of files*). Namely, *Kumar* discloses its

malware classifiers can “be trained” with “labeled files.” *Id.*, [0047]. In an example, *Kumar* discloses classifying Rbot malware by training Rbot binary classifiers with features from “fixed or randomly selected subset of 50,000 labeled Rbot files.” *Id.* To train the classifier with these files, as explained above, “a feature selection algorithm attempts to choose the 3,000 ‘best’ features per family, and 12,000 ‘best’ features for the generic malware and benign families (where ‘best’ may be determined according to an appropriate ranking metric).” *Id.*, [0055]. In this way, “a ranked list of potential features for each family is computed, and used to choose the top K (3000, 12000) features for the family” such that these features “discriminate the one family from other malware and benign file classes (e.g., other malware families, generic malware family, other generic families, and generic benign family).” *Id.*, [0056]. “The output of the training system comprises a classifier model 232 (e.g., a set of weights or parameters), which can be used to predict if an unknown report was generated by a particular malware family, a generic malware class, a particular benign family, or a generic benign class.” *Id.*, [0052].

86. Based on this disclosure, a PHOSITA would have understood that *Kumar*’s system receives files as input. These files are processed to be mapped to the list of static data points. And then useful features are derived from these lists by the training model. In this way, the training data comprises files known to fit in one of the plurality of categories of files.

87. I note that *Kumar*'s training process is consistent with the embodiments disclosed in the '869 Patent. For example, the '869 Patent explains that "[t]o classify executable files...knowledge component 102 collects static data on a large variety of executable files (e.g., PE files)." '869 Patent, 4:29-31. This static data "yields a knowledgebase that may be used to periodically train a classifier." *Id.*, 4:39-43; *see also id.*, 7:34-42 (describing "the information used to train/re-train the learning classifier" including "static data on the robust collection of executable files"), 11:60-67 (describing method 300, including operation 302 which "builds the knowledge base from the data collected and evaluated related to known" files). For the reasons above, *Kumar*'s training process is consistent with these disclosures.

88. After training, in a malware detection phase, the features extracted from the unknown file (i.e., the plurality of static data points identified in Claim 1(a) and 1(d)) are transmitted to the backend as a telemetry report and processed by detection logic 102:

[I]n this example step 506 is performed at the client to extract the features and any other characteristics data from the file, which is placed into a report. Step 508 sends the report in a query to the backend service...Step 515 represents receiving the report (or file if sent via step 514) from the frontend client at the backend service, and classifying the report as indicative of malware or not based upon the data (of file content) therein.

Id., [0067]; *see also id.*, [0026] (noting detection logic 102 includes "trained telemetry report classifiers"), Fig. 5.

89. In this malware detection phase, a PHOSITA would have understood or found obvious that the extracted telemetry data (i.e., *plurality of static data points*) is converted into feature vectors for processing by “one or more trained telemetry report classifiers that determine, in part, whether the features and characteristics are indicative of malware[.]” *Kumar*, [0026].

90. For example, *Kumar* describes that “[t]o this end, the backend malware detection logic 102 includes one or more trained telemetry report classifiers that determine, in part, whether the features and characteristics are indicative of malware, or more specifically (in one implementation) a particular family of malware.” *Kumar*, [0026]. *Kumar* provides several examples of classifiers which may be employed in its system, such as “logistic regression, naive Bayes, support vector machine, neural network, boosted decision tree, other boosted algorithms, ensemble classifiers, hierarchical classifiers and/or the like.” *Id.*, [0027]. A PHOSITA would have understood that several of these classifiers require feature vectors as input to process feature data. As just one example, a support vector machine (SVM) utilizes features vector as input for processing.

91. Indeed, it was well-known that SVMs use feature vectors as input. For instance, U.S. Patent Application No. 2013/0326625 to Anderson et al. (“*Anderson*”) (Ex. 1016) describes a process for generating feature vectors for use in an SVM classifier:

In any of the examples herein, data sources can be represented using data representations. In some implementations, the file information collected for a file information data source can be represented as a feature vector (e.g., a feature vector of length seven or other length) where the statistics for a feature of the data source corresponds to features of the feature vector.

Anderson, [0208]; [0181] (“In some implementations, the classification of the program can include a classification of malware or non-malware. In some implementations, a classifier, such as a kernel-based classification algorithm (e.g., a support vector machine, a Gaussian process, or the like), is trained using kernels generated using one or more combinations of representations of dynamic, static, and/or file information data sources of programs from a data set of malware and non-malware programs”).

92. Moreover, U.S. Patent No. 8,266,698 (“*Seshardi*”) similarly describes inputting feature vectors to an SVM. *Seshardi* (Ex. 1017), 4:11-17 (“The malware security module 116 applies the malware classifier 130 to the feature vectors in order to generate malware scores that indicate the likelihood that the suspicious software applications are malware”), 9:61-10:3 (describing “support vector machine” as classifier), Figs. 5-7. Thus, using a feature vector as input to a SVM was well-known in the art.

93. Additionally, U.S. Patent No. 9,489,514 to Mankin et al. (“*Mankin*”) (Ex. 1018) describes the same. *Mankin* describes that its “machine learning system can include at least one of: a support vector machine, a decision tree, and a k-nearest-

neighbor classifier.” *Mankin*, 2:24-26. And notably, *Mankin* describes “[a]t block 208, the method trains a machine learning system with the classified feature vectors.” *Mankin*, 8:21-22.

94. Furthermore, *Kumar* already discloses generating feature vectors from static data points to train a classifier, such that a PHOSITA would have understood that during detection, feature vectors would similarly be generated in the same way to facilitate classification. *Kumar*, [0052] (“[A] preprocessing step (not shown) is usually needed to convert the raw telemetry XML data to the actual **feature vector**”), [0055] (“convert the raw data recovered by the system to the actual **feature vector**”).

95. A PHOSITA would have been motivated to use a support vector machine (SVM) as the classifier in *Kumar*’s threat detection system to appreciate the well-known benefits of SVMs compared to other types of classifiers. First, SVMs were well known and widely used in computer/network security and threat detection systems. *See, e.g., Anderson, Seshardi, and Mankin* above. Moreover, “Intrusion Detection Using The Support Vector Machine Enhanced With A Feature--Weight Kernel” by Songlun Zhao et al. (“*Zhao*”) (Ex. 1019) supports that using an SVM as a classifier was well known and further describes advantages of using an SVM classifier, such as enhanced precision for detecting intrusions and making classifications. *Zhao*, Abstract (“using SVM as a classic pattern recognition tool,

have been widely used in intrusion detection”), 53 (describing high precision in SVMs for detecting intrusions in datasets), 65 (“[t]he conventional SVM can classify the dataset very well”). Additionally, SVMs provide for “good generalization,” a solution to the high dimensionality concerns, and overall better performance than other traditional AI techniques. *Id.*, 2-3, 13-14, 18-21.

96. As another example, a PHOSITA would have been motivated to use an SVM as a classifier for an SVMs time-saving abilities. For instance, “Intrusion Detection Using Neural Networks and Support Vector Machines” by Srinivas Mukkamala et al. (“*Mukkamala*”) (Ex. 1020) describes that “[i]t is observed that both the neural networks and SVMs deliver highly accurate results (greater than 99% accuracy on testing set) and show compatible level of performance. The training time for SVMs is significantly shorter (17.77 sec vs. 18 min), an advantage that becomes rather important in situations where retraining needs to be done quickly (e.g., when new attack patterns are discovered). The running time of SVMs is also notably shorter.” *Mukkamala*, 1706.

97. As another example, a PHOSITA would have been motivated to use a SVM as a classifier due to SVM’s efficient performance metrics. For instance, “Classification of Malicious Domain Names using Support Vector Machine and Bi-gram Method” by Nhaou Davuth et al. (“*Davuth*”) (Ex. 1021) describes:

SVMs are powerful learning approach for solving two-class pattern recognition ... The theoretical conclusion is that SVMs are suitable for

textual classification because SVM performs well with high dimensional feature spaces, few irrelevant features (dense concept vector), and sparse instance vectors.

Davuth, 52. *Davuth* teaches “[a]nyways, if we compare with other algorithms such as Naïve Bayesian or C5.0, we found that SVM gave an effective result and is suitable to classify the two-class examples with a high dimensional space.” *Davuth*, 57.

98. Further, “Malware behavioral detection and vaccine development by using a support vector model classifier” by Ping Wang et al. (“*Wang*”) (Ex. 1022) confirms these known benefits, describing:

Solving classification problems by using an SVM classifier (SVC) guarantees few classification errors caused by maximizing the generalisation ability of learning incorporating the Lagrange multiplier optimization algorithm [8]. Generally, the results from using SVM classification algorithms are more accurate than those from other ML approaches involving non-optimised search methods, such as artificial neural networks, least squares, k-nearest neighbour, Bayesian probability, and classification and regression trees [9][.]

Wang, 1013.

99. In sum, a PHOSITA would have been motivated to use an SVM as a classifier out of the list of possible classifiers *Kumar* describes at least due to the high-performance metrics, time-saving ability, solution to high dimensionality concerns, and precision/accuracy when using an SVM as a classifier. And a PHOSITA would have had a reasonable expectation of success using the SVM as the classifier. *Kumar* expressly describes that its system is compatible with SVMs.

Moreover, SVMs are well-known classifiers for threat detection with widely appreciated benefits.

- (g) *[1(e)(i)] wherein one or more features of the feature vector are selectively turned on or off based at least in part on evaluation of whether a value of one of the plurality of static data points is within a predetermined range; and*

100. As I discussed in Claim 1(e) above, *Kumar* discloses or renders obvious generating a feature vector from the extracted features during a malware detection phase. These extracted features are shown in the table below:

Feature	Description
File Name	Name of the PE file
Original File Name	File name in the original report
File Name Matches Original?	Whether the file name in this report matches the file name in the original report
File Type	Type of file it is
Signer Name	Organization that signed the file
Signing Authority	Certificate authority that issued the signature
Signature Type	File signed or not; if signed, whether the signature is legitimate or invalid
Description	Description of the file in the header
Organization	Manufacturer of the binary file
Version	Version number of the binary file
Packed	Whether the file is packed
LS Hash	Locality sensitive hash
Behavior Feature	Represents the simple behavior that caused the report to be generated

Kumar, [0028].

101. *Kumar* additionally discloses a process for turning off features during evaluation. Namely, *Kumar* discloses that “**one or more other features** may be used

to indicate if a particular attribute is blank or null.” *Id.* For example, in the table above, if the “organization” feature “is null, a boolean feature is set to true.” *Id.*

102. A PHOSITA would have understood that a separate boolean feature associated with the “organization” attribute set to “true” indicates that the malware file did not contain “organization” data representing the “[m]anufacturer of the binary file.” By setting the boolean feature to “true,” *Kumar’s* system discards (i.e., *turns off*) the “organization” feature such that only the Boolean feature is input to the classifier to indicate the absence of organization data. A PHOSITA would have understood this boolean feature would be used in lieu of the “blank or null” organization feature because it would be redundant to introduce both a blank/null feature state *and* a boolean feature indicating this feature was blank/null for classification. And it was known to reduce redundancies in computing analyses, particularly where prompt analytics are important, such as in threat detection.

103. In another example, if *Kumar’s* system extracted the static telemetry data identified in the table above and the “Signature Type” was blank/null (meaning the file was not signed) then a boolean value for the “Signature Type” feature would similarly be set to true, turning off the “Signature Type” feature and introducing only the boolean feature for classification. Specifically, in this example, because there was no “Signature Type” data in the file being analyzed, other associated features would similarly have boolean values set to true and would therefore be turned off.

This would include the “Signer Name” and “Signing Authority” features, such that each of these features would have Boolean values set to true, thereby turning these features off given that the file was not signed. Indeed, an unsigned file has no data concerning what organization signed the file (Signer Name), nor what certificate authority issued the signature (Signing Authority).

“within a predetermined range” (PO’s interpretation)

104. Accordingly, in each of these examples, *Kumar’s* system is selectively turning off one or more features in a feature vector based on a value of one of the static data points being within a predetermined range—i.e., whether or not the features have any value, or are blank/null. I understand that this is consistent with PO’s proposed interpretation of the claim language it advanced in parallel litigation.

105. It is my understanding that PO suggested the plain and ordinary meaning of “within a predetermined range” is not limited to “a numerical range or a subset of a ‘particular scale’ with at least a lower limit or an upper limit, but not necessarily both.” Ex. 1007 (*Responsive Markman Brief, Dkt. 304*), 59-60. PO also argued the “assumption that the predetermined range has to be a numerical range is [] inconsistent with the ‘844 specification.” Ex. 1009 (*Responsive Markman Brief, Dkt. 98*), 66-67 (citing overlapping disclosure in the parent ’844 Patent concerning the “legal information” example also found in the ’869 specification). PO continued that “predetermined ranges are not restricted to numeric values on a spectrum...the

‘predetermined range’ in the case of ‘legal language’ in the body of the executable file is that if such language is present, then the learning classifier allows certain other features of the vector to be turned off. Such a determination does not require three numeric data points, but rather the ‘predetermined range’ in this disclosed instance is whether or not the executable file includes legal information.” *Id.*, 67-68. In support, PO pointed to the disclosed example in which the classifier “may detect that the file does not contain ‘legal information’ in the file.” ’869 Patent, 7:43-64. “In an example where the learning classifier detects the presence of legal information does not exist in the executable file, this may trigger the learning classifier to... turn off other features related to an evaluation of the ‘legal information.’” *Id.* I have been asked to apply PO’s interpretation. I express no opinion as to the correctness of this interpretation.

106. Analogous to PO’s examples and discussion, *Kumar* detects the absence of, e.g., organization or signature type data, and turns off these features and other related features and sends only the boolean features for classification. This is also consistent with known in the art processes. For example, it was generally known in the art to turn off features of a feature vector during file classification. “Mal-ID: Automatic Malware Detection Using Common Segment Analysis and Meta-Features” by Gil Tahan et al. (“*Tahan*”) (Ex. 1005) describes using each segment’s entropy value to selectively turn on or off the associated segment’s features (i.e., the

3-grams associated with the segment) based on an evaluation of whether the entropy value is within a predetermined range. *Tahan*, 958.

“within a predetermined range” (District Court’s construction)

107. I understand that the District Court construed “within a predetermined range” with respect to the parent ’844 Patent to carry its plain and ordinary meaning, where “range” requires a “numerical range or a subset of a ‘particular scale’ with at least a lower limit or an upper limit, but not necessarily both.” Ex. 1010 (*Markman Order*), 18. I also understand that in claim construction briefing regarding the ’869 Patent, PO acknowledged the court’s construction, but expressly preserved its rights to appeal based on the arguments it advanced with respect to the ’844 Patent. Ex. 1007 (*Responsive Markman Brief, Dkt. 304*), 59-60.

108. If the PO interprets the District Court construction broad enough to encompass a binary situation such that the existence or non-existence of a value defines the lower or upper limit, *Kumar* satisfies this interpretation as well for all of the reasons I described above regarding turning off features relating to organization, Signature Type, Signer Name, Signing Authority, etc.

109. Alternatively, to the extent the District Court construction requires something more than simply the existence or absence of a value, it is my opinion that it would have been obvious to modify *Kumar*’s methods to perform this step

based on the teachings of *Tahan*, which satisfies this limitation under any construction.

110. *Tahan* (Ex. 1005) discloses a similar static malware detection method that “operates by analyzing short segments extracted from the file examined.” *Tahan*, 957. *Tahan*’s segments comprise “a number of 3 grams depending on the length of the segment[.]” *Tahan*, 957. Three features can be “derived for each segment: Spread, MFG, and Entropy.” *Tahan*, 957. These features “are in fact meta-features as they are used to represent features of features (features of basic 3-grams).” *Tahan*, 957.

111. During a malware detection phase, *Tahan*’s method uses each segment’s entropy value to selectively turn on or off the associated segment’s features (i.e., the 3-grams associated with the segment) based on an evaluation of whether the entropy value is within a predetermined range. For example, *Tahan*’s algorithm gets, “two parameters EntropyLow and EntropyHigh.” *Tahan*, 958. *Tahan*’s system then “check[s] if the entropy is smaller than EntropyLow threshold or entropy is larger than Entropy-High...[and] [i]f so then discard[s] the segment and continue[s] segment iteration.” *Tahan*, 958. A PHOSITA would have understood that in *Tahan*’s method, if *Tahan* determines that an entropy value falls outside the range defined by the EntropyLow and EntropyHigh thresholds, the segment, and the 3-gram features associated with the segment, are discarded.

112. In my opinion, pursuant to *Tahan's* teachings, it would have been obvious to a PHOSITA to configure *Kumar's* detection logic to selectively turn on/off features in a feature vector based on numerical values of those features. Such a modification would further benefit classification here, ignoring certain features that would be unhelpful for classification based on specific numerical values for the file being analyzed. This would reduce redundant processing and optimize over the most relevant features for classification.

113. For example, *Kumar* discloses numerical static telemetry data points which are extracted from a file and used as features for the classifier, such as “Version” (“Version number of the binary file”). *Kumar*, [0028], [0032]. As I describe below, this “Version” feature data is numerical data.

114. *Kumar* also explains its disclosed features are “example features derived from the telemetry data.” *Id.* [0028], [0055] (“the features described herein are only examples”). *Kumar* discloses that “the classifier is not limited to these types of features, as others...may be used instead of or in addition to some or all of the ones exemplified herein.” *Id.* Based on this disclosure, a PHOSITA would have found it obvious to extract other known numerical features to benefit classification, such as a file size. For example, “FIRMA: Malware Clustering and Network Signature Generation with Mixed Network Behaviors” by M. Zubair Rafique (“*FIRMA*”) (Ex. 1023) describes a malware detection system that uses a file size as

an extracted feature for classification. *FIRMA*, 151 (describing extracting the following feature vector, [fid, rid, proto, msg, sport, dport, sip, dip, **dsize**, endpoint, ptree], where “dsize is the size in bytes” of the malware file).

115. Based on the teachings of *Tahan*, it would have been obvious to modify *Kumar*’s detection logic to selectively turn on/off numerical features, such as file size and version features, by comparing these values to a threshold, as disclosed in *Tahan*. For example, with respect to file size, it was known in the art that certain numerical features are irrelevant for classification when their values fall below a certain range. *FIRMA*, 150 (noting that 4 KB is “the minimum size of a working malware” that has been observed). Based on these known teachings, a PHOSITA would have understood that file size features extracted by *Kumar*’s detection logic with numerical values under 4 KB may be discarded. To discard these features, *Kumar*’s system would be modified to set boolean values to “true” for any file size features below 4 KB. Thus, these features would be “turned off” based on a “numerical range or a subset of a ‘particular scale’ with at least a lower limit or an upper limit.” A PHOSITA would have been motivated to make this modification to appreciate the known benefits described above, including reducing redundant processing and optimizing over the most relevant features for classification. And a PHOSITA would have had success making this modification as *Kumar* already expressly contemplates using other features for classification; file size was a known

feature used in malware classification systems; and modifying *Kumar* to extract this additional feature and turn it off based on known unusable file sizes pursuant to *Kumar*'s and *Tahan*'s teachings would have required nothing more than the ordinary skill of a PHOSITA.

116. With respect to a version number, a PHOSITA would have also found it obvious, based on *Tahan*'s teachings, to turn off "Version" feature data based on a numerical range. For example, as I explained above, both *Kumar* and *Tahan* disclose deriving entropy values for feature data. *See, e.g., Kumar*, [0035] (describing entropy values with filenames); *Tahan*, 957 (describing entropy for segments comprising n-gram features). A PHOSITA would have found it obvious to derive entropy values for other feature data, such as file "Version" data. This would further enhance classification, optimizing the features that are considered at runtime by enabling certain extracted version data features to be discarded or ignored based on the derived entropy values. Like above, a PHOSITA would have had success making this modification as it would have been well within the skill of a PHOSITA to modify the programming parameters in *Kumar*'s system with no change to the already-present hardware.

117. More specifically, in this modified system, entropy values would be derived for feature data relating to the "version" of a file, and then when version entropy values are outside a predetermined range, the features related to these

extracted version data points would be discarded pursuant to *Kumar's* and *Tahan's* teachings. As described in *Tahan*, the predetermined range is the EntropyLow threshold and the Entropy-High threshold. This process would optimize *Kumar's* classification process and discard irrelevant “version” data with too low or too high entropy values during runtime, reducing the number of features considered based on the specific file being evaluated. Too low entropy for “version” data (e.g., entropy values below EntropyLow) would indicate little (or no) difference in version data from multiple instances of this file, and the classifier would discard this data point as irrelevant to classification. Similarly, if the entropy values exceed EntropyHigh, this would indicate too much variability such that the version data is not a good measure of data within the file and is similarly discarded. Therefore, in *Kumar's* system, when version feature data values are below EntropyLow or above EntropyHigh, a boolean feature would be set to true, discarding the original feature and sending only the boolean feature to the classifier. I note that high entropy values in *Kumar* indicate that a specific file may be packed—i.e., which tends toward malware. However, in *Tahan*, entropy values that are exceedingly high are discarded. Given these disclosures, *Kumar's* and *Tahan's* processes would interoperate seamlessly, discarding features with exceedingly high entropy values which are unusable for classification.

118. I note that both “version” and “file size” features are “numerical” consistent with the ’869 Patent specification shown in Table 1.1 below.

TABLE 1.1			
Numeric values	Nominal values	Strings/Byte sequences	Boolean values
File size	initialize	Comments	Address Of Entry Point Anomaly
linker version	Un-initialize	company name	Image Base Anomaly
code size	entry point	file description	Section Alignment Anomaly
OS version	subsystem	internal name	Size Of Code Mismatch Anomaly
image version	file subtype	legal copyright	Low Import Count Anomaly
subsystem version	language	original file	Entry Point Anomaly
file version number	file flags masks	private build	certificate Validity
product version number	file flags	product name	Certificate Exception
size of heap	file OS	special build	Code Characteristics Anomaly
size of stack	file type	product version	Code Name Anomaly
size of image	machine type	file version	Count Anomaly
PE header time	PE type	package code	Data Characteristics Anomaly
Section Entropy	section counts	product code	Data Name Anomaly
Sections count	DLL count	export DLL name	Export Exception
	DLL functions	assembly version	Large Number of DLLs Anomaly
	data directory	Certificate Issuer	flag DLL Name Anomaly
	export count	Certificate Subject	Number of Functions Anomaly
	Earliest Data Byte	Imports	Function Name Anomaly
	resources	Exports	PE Header Anomaly
	resources language	Section Names	High Section Count Anomaly
	resource Encoding	Non-resource section strings	PE Magic Validity
	resource code page		Resource Exception
	resource size		VR Code Ratio Anomaly
	DLL characteristics		Import Exception

’869 Patent, Table 1.1 (listing “linker version,” “OS version,” “image version,” “subsystem version,” “**file version number**” along with “**file size**” and “code size” under the “Numeric values” category).

119. Additionally, it would have been obvious to a PHOSITA to configure *Kumar*’s detection logic to selectively turn on/off features, such as a filename, in a feature vector based on the feature’s entropy value being within a predetermined range. As discussed above, *Kumar* describes extracting static data points from files as shown below:

Feature	Description
File Name	Name of the PE file
Original File Name	File name in the original report
File Name Matches Original?	Whether the file name in this report matches the file name in the original report
File Type	Type of file it is
Signer Name	Organization that signed the file
Signing Authority	Certificate authority that issued the signature
Signature Type	File signed or not; if signed, whether the signature is legitimate or invalid
Description	Description of the file in the header
Organization	Manufacturer of the binary file
Version	Version number of the binary file
Packed	Whether the file is packed
LS Hash	Locality sensitive hash
Behavior Feature	Represents the simple behavior that caused the report to be generated

Kumar, [0028]. One of the static data points described in *Kumar* is a “File Name.” In some embodiments the letters in the filename in *Kumar*, “can be converted into a set of trigrams where each trigram index represents three consecutive letters in the file name.” *Kumar*, [0052]. Further, *Kumar*, like *Tahan*, describes deriving meta-features (features of features). *Kumar*, [0035]; *Tahan*, 957. Namely, *Kumar* discloses deriving the “entropy of the observed filenames” to be “used as a feature” for classification, which represents “[s]ome measure of the difference in the file names” across the same file running on different computers. *Kumar*, [0035]. Determining these entropy values is consistent with the ’869 Patent’s disclosures. The ’869 Patent lists “Section Entropy” as a numerical static data feature, similar to *Kumar* and *Tahan*.

TABLE 1.1			
Numeric values	Nominal values	Strings/Byte sequences	Boolean values
File size	initialize	Comments	Address Of Entry Point Anomaly
linker version	Un-initialize	company name	Image Base Anomaly
code size	entry point	file description	Section Alignment Anomaly
OS version	subsystem	internal name	Size Of Code Mismatch Anomaly
image version	file subtype	legal copyright	Low Import Count Anomaly
subsystem version	language	original file	Entry Point Anomaly
file version number	file flags masks	private build	certificate Validity
product version number	file flags	product name	Certificate Exception
size of heap	file OS	special build	Code Characteristics Anomaly
size of stack	file type	product version	Code Name Anomaly
size of image	machine type	file version	Count Anomaly
PE header time	PE type	package code	Data Characteristics Anomaly
Section Entropy	section counts	product code	Data Name Anomaly
Sections count	DLL count	export DLL name	Export Exception
	DLL functions	assembly version	Large Number of DLLs Anomaly
	data directory	Certificate Issuer	flag DLL Name Anomaly
	export count	Certificate Subject	Number of Functions Anomaly
	Earliest Data Byte	Imports	Function Name Anomaly
	resources	Exports	PE Header Anomaly
	resources language	Section Names	High Section Count Anomaly
	resource Encoding	Non-resource section strings	PE Magic Validity
	resource code page		Resource Exception
	resource size		VR Code Ratio Anomaly
	DLL characteristics		Import Exception

'869 Patent, Table 1.1.

120. When *Kumar* is modified by *Tahan*, *Kumar's* threat detection system would evaluate the filename entropy values to determine whether these values fall within a predetermined range. The predetermined range is the EntropyLow threshold and the Entropy-High threshold. This process would optimize *Kumar's* classification process and discard irrelevant filenames with too low or too high entropy values during runtime, reducing the number of features considered based on the specific file being evaluated.

121. A PHOSITA would have understood “discarded” filenames are “turned off” within the meaning of the '869 Patent. Indeed, the '869 Patent describes:

As an example, the learning classifier can adaptively set features of a learning classifier based on the static data points extracted for a file. For example, ranges of data can be identified for static data points that may

enable the learning classifier to controllably select (e.g., turn on/off) features for evaluation by learning classifier. **In one instance where a static data point extracted for evaluation is a file size (e.g., numeric value), the learning classifier may turn on/off features for evaluation based on whether a the file size of the file is within a certain range.** In another example, the learning classifier may detect that the file does not contain “legal information” in the file. Legal information may be any identifying information indicating data that conveys rights to one or more parties including: timestamp data, licensing information, copyright information, indication of intellectual property protection, etc. In an example where the learning classifier detects the presence of legal information does not exist in the executable file, this may trigger the learning classifier to adaptively check for additional features that might be indicative of a threat or malicious file as well as turn off other features related to an evaluation of the “legal information.” The learning classifier component 104 utilizes programming operations or machine-learning processing to train/re-train the learning classifier to uniquely evaluate any file/file type.

’869 Patent, 7:43-64 (emphasis added).

122. A PHOSITA would have recognized that it was known in the art to use thresholds in malware detection systems to discard irrelevant features. For example, “Classification of Malicious Domain Names using Support Vector Machine and Bi-gram Method by Nhauo Davuth et al. (“*Davuth*”) (Ex. 1021) describes: “In our experiment, we set some threshold values to remove useless features. We assume that f_1 is the frequency of each feature for legitimate domains and f_2 is the frequency of each feature for malicious domains, and we want to select the features that have frequency bigger than 100[.]” *Davuth*, 55.

123. As another example, U.S. Patent Publication No. 2008/0319932

to Wen-tau Yih et al. (“*Yih*”) (Ex. 1024) describes:

Depending on the threshold picked, the classifier will optimize on a different type of data, and thus optimize for a different false positive rate. FIG. 2C illustrates the training data employed by the second stage classifier when training data that is classified below the threshold by the first stage classifier is ignored. In other words using the spam e-mail filtering example, eliminating all of the easy to classify good e-mails (□ squares in the bottom right) and hard to classify spam e-mails (x's in the bottom right) by picking a threshold for the first-stage classifier. FIG. 2D illustrates the second-stage classifier that is learned using the training data from FIG. 2C.

Yih, [0040].

124. Therefore, in operation, in *Kumar*'s methods as modified by *Tahan*, a specific file with filename entropy value below EntropyLow would indicate (little (or no) difference in filenames from multiple instances of this file, and the classifier would discard this data point as irrelevant to classification. On the other hand, if the entropy values exceed EntropyHigh, this would indicate too much variability such that the filename is not a good measure of data within the file and is similarly discarded.

125. *Tahan*'s process is not duplicative of *Kumar*'s feature selection process. *Kumar*'s feature selection process occurs during training, optimizing the specific features which the system deems best for categorization. However, this feature selection process does not optimize features on a file-specific basis. Using *Tahan*'s

processes, irrelevant features of the specific file being analyzed are discarded at runtime, further optimizing the overall classification.

126. Further, a PHOSITA would have had a reasonable expectation of success in making these modifications. *Kumar* already discloses that its detection logic can be enhanced to “use larger resource limits, more signatures, more in-depth processing, and so forth.” *Kumar*, [0024]. Adding an additional processing routine would have required only programming changes to *Kumar*’s already present detection logic with the tradeoff of optimizing classification on a “file specific” basis. Such programming changes would have been well within the skill of PHOSITA during the relevant timeframe.

- (h) *[1(f)] providing the generated feature vector to one or more support vector machines to build a probabilistic model that indicates whether the executable file fits into one of the categories of files.*

127. As I discussed above, *Kumar* discloses using a support vector machine to classify the feature vectors generated during malware detection. *See, e.g., Kumar*, [0027]. Indeed, *Kumar* discloses an exemplary list of classifiers operable with its disclosed threat detection system, including a “support vector machine” (SVM):

In one implementation of the telemetry classifier, a multiclass classifier (e.g., logistic regression, naïve Bayes, **support vector machine**, neural network, boosted decision tree, other boosted algorithms, ensemble classifiers, hierarchical classifiers and/or the like) may be used to assign an unknown telemetry report to a particular malware family, with a generic malware family representing other families, a particular benign group of applications,

and/or the generic benign class representing any remaining benign applications.

Kumar, [0027], [0039]. For the same reasons identified above, a PHOSITA would have been motivated to use an SVM as the classifier in *Kumar's* threat detection system to appreciate the known benefits and advantages of using the same compared to other known classifiers.

128. In *Kumar's* SVM classification techniques, the generated feature vector is input to the SVM, and this SVM builds a probabilistic model that indicates whether the executable file is malware or benign (i.e., *fits into one of the categories of files*). In this regard, *Kumar* discloses that the backend malware detection logic may include “the results of a file based malware classifier which **assigns a probability to the file that it belongs to a particular malware family, the generic malware family, a particular benign family, or the generic benign family.**”

Kumar, [0037]. An exemplary probabilistic model is shown below:

Maximum Category Probability	Category Probability	First substring	Second substring
P_{Benign}	>0.9	“Benign_4”	“”
P_{Benign}	$0.7-0.9$	“Benign_3”	(The empty string)
P_{Benign}	$0.5-0.7$	“Benign_2”	
P_{Benign}	$0.333-0.50$	“Benign_1”	
$P_{Inconclusive}$	>0.333	“Inconclusive”	
$P_{Malware}$	$0.333-0.50$	“Malware_1”	The most likely
$P_{Malware}$	$0.5-0.7$	“Malware_2”	family string
$P_{Malware}$	$0.7-0.9$	“Malware_3”	(e.g. Rbot)
$P_{Malware}$	>0.9	“Malware_4”	

Kumar, [0043]. *Kumar* discloses that “[o]ther inconclusive results can be computed and returned for other types of classifiers including naïve Bayes, support vector machine, neural network, boosted decision tree, other boosted algorithms, ensemble classifiers, hierarchical classifiers and/or the like” as well. *Kumar*, [0045]. Based on *Kumar*’s discloses, a PHOSITA would have understood or found obvious that the SVM classifier would build a probabilistic model, indicated above, to categorize the detected executable file—assigning the file to a particular category selected from the subset of benign, inconclusive, or malware.

2. *Claim 2 – The computer-implemented method according to claim 1, wherein the plurality of static data points are extracted without decrypting or unpacking the executable file.*

129. As I explained in my opinions above, *Kumar* discloses that after identifying a potential malware file, antimalware client component 116 “extract[s]” features and characteristics (i.e., *plurality of static data points*) from the identified executable file to send to backend service 104 for analysis. *See above*, Claim 1(d). While *Kumar*’s system can unpack a file to obtain certain data points (e.g., *Kumar*, [0055] “unpacked strings”), a PHOSITA would have understood or found obvious that *Kumar*’s other exemplary extracted features can be extracted without decryption or unpacking. For instance, these features are identified below:

Feature	Description
File Name	Name of the PE file
Original File Name	File name in the original report
File Name Matches Original?	Whether the file name in this report matches the file name in the original report
File Type	Type of file it is
Signer Name	Organization that signed the file
Signing Authority	Certificate authority that issued the signature
Signature Type	File signed or not; if signed, whether the signature is legitimate or invalid
Description	Description of the file in the header
Organization	Manufacturer of the binary file
Version	Version number of the binary file
Packed	Whether the file is packed
LS Hash	Locality sensitive hash
Behavior Feature	Represents the simple behavior that caused the report to be generated

Kumar, [0028]. A PHOSITA would have understood that a file name, original file name, file name matches, file type, signer name, etc. and much of the other data provided in the table above is extracted from PE file headers, which are not packed or encrypted.

130. Additionally, if the identified executable is not packed and encrypted, *Kumar* would extract features without unpacking or decryption. Indeed, *Kumar* discloses that its system can obtain file strings “directly from the file” without unpacking for files that are not packed or encrypted. *Kumar*, [0039]. For packed and/or encrypted files, a PHOSITA would have understood *Kumar* would extract many of the above features without unpacking or decryption. Indeed, a PHOSITA would have understood static features are those obtained by simply investigating the binary contents of a file. In contrast, dynamic features require executing the file in a

sandbox monitoring system. “Comparative Analysis of Combinations of Dimension Reduction and Data Mining Techniques for Malware Detection,” by Proceso L. Fernandez Jr. et al. (“*Fernandez*”) (Ex. 1025), 27. This is consistent with the ’869 Patent’s disclosures, such that these same features identified in Table 1.1. (e.g., file name, file type, etc.), can be extracted “regardless of whether or not they are encrypted and/or compressed.” ’869 Patent, 6:45-47

131. Alternatively, a POSITA would have found it obvious to use a known static feature extraction tool, such as “PE Dump” to obtain PE file features, such as filenames without unpacking or decryption. A PHOSITA would have recognized several known in the art static feature extraction tools which would facilitate this processing in *Kumar*’s system. For example, *Fernandez* provides a “list of tools used for static feature extraction[,]” identified below:

- PE Dump - used to dump the PE file features of the file
- Bintext - used to dump the readable ANSI and UNI-CODE text strings in the file
- Trend Micro VSDT tool - extracts the resource information from the file
- Trend Micro ScanPacker tool - extracts the packer in-formation from the file
- Trend Micro Entropy tool - extracts the entropy computations on the file sections

Fernandez, 27. And *Fernandez* explains these tools obtain the following types of information:

- Windows Libraries and Functions Used
- Extracted Significant Readable Text Strings

- File Structure Details (Portable Executable File header, section tables, resource table, Import/Export tables)
- Compression Details (Entropy, Packer Type/Layer)
- Malware-related URL's in the code

Id.

132. A PHOSITA would have been motivated to use any of these known static feature extraction tools in *Kumar's* system to extract static data features, as *Kumar* already contemplates. Using known tools would successfully extract the necessary feature data for *Kumar's* classification process. Incorporating any of these tools is simply combining prior art elements according to known methods to predictably yield a system which successfully extracts static data features for subsequent malware detection/processing.

3. *Claim 3 – The computer-implemented method according to claim 1, wherein the one or more support vector machines builds the probabilistic model by performing data analysis and pattern recognition on the one or more feature vectors.*

133. As described above in Claim 1(f), *Kumar's* SVM builds a probabilistic model. *Kumar* discloses that its probabilistic model is built from the following function:

$$s[j, n] = \sum_{f=0}^{F-1} w(f, j)x(f, n) + b(j)W(i, j)$$

Kumar, [0040]. Here, “ $w(f,j)$ ” is the binary classifier weight for feature f for the j th family classifier...[t]here are F_{total} features, and $x(f,n)$ is the f th feature value for

the n th unknown file to be evaluated.” *Kumar*, [0040]. The classifier computes 102 “1 versus the rest” linear classifiers (100 specific malware families, 1 generic malware family, and 1 benign family), then “computes the overall probability for the n th file.” *Id.* If the classifier is set to include an “inconclusive” result, three separate output probabilities are used for the three states, “Malware,” Benign,” and “Inconclusive”:

$$\begin{aligned} P_{benign}(n) &= e^{s[Benign,n]} / (\sum_k e^{s[k,n]} + 1) \\ P_{Malware}(n) &= \sum_{j \neq Benign} e^{s[j,n]} / (\sum_k e^{s[k,n]} + 1) \\ P_{Inconclusive}(n) &= 1 / (\sum_k e^{s[k,n]} + 1) \end{aligned}$$

Kumar, [0041].

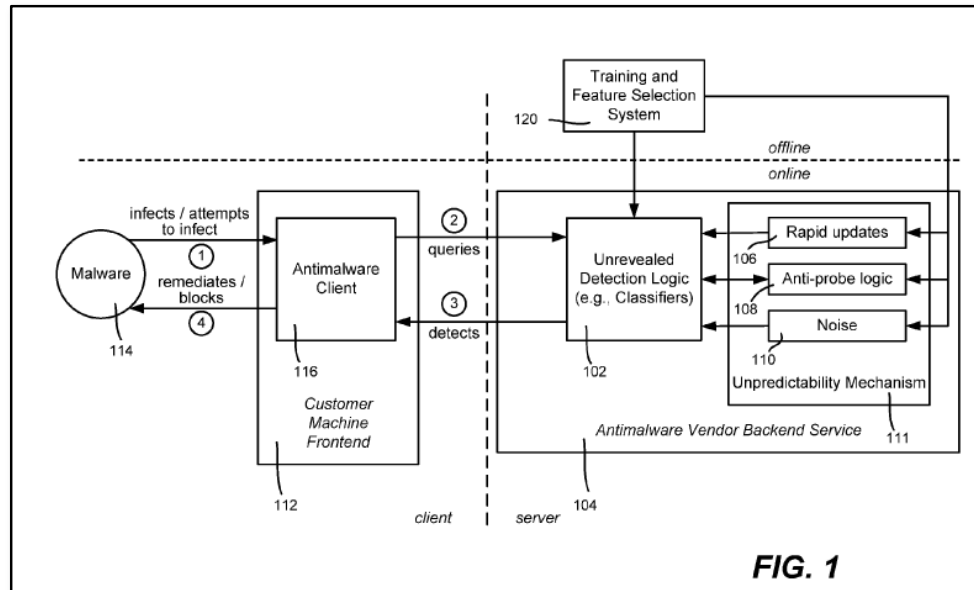
134. A PHOSITA would have understood that *Kumar*’s functions above perform data analysis and pattern recognition on one or more feature vectors (e.g., applying weights to feature “ f ”) to build its probabilistic model. For example, it was well-known that SVMs “are supervised learning models associated with learning algorithms and used to **analy[z]e data** and **recogni[z]e patterns**.” *Wang*, (Ex. 1022), 1013; *see also* U.S. Patent No. 8,418,249 to Nucci et al. (“*Nucci*”) (Ex. 1026), 2:25-28 (which describes that “[a] support vector machine (SVM) is a set of supervised learning methods that analyze statistically related data items and recognize patterns for classification and regression analysis”). As another example,

U.S. Patent No. 8,161,548 to Wan (“*Wan*”) (Ex. 1027) describes a “malware classifier apparatus as recited in claim 16 wherein said pattern classification algorithm is based on the support vector machine (SVM) algorithm.” *Wan*, claim 17. Further *Wan* describes that in “step 520 the pattern classification algorithm (in this embodiment, the SVM Light software) accept as input the extracted values and by use of the trained model outputs a classification label 150.” *Wan*, 38-41. Thus, a PHOSITA would have understood that *Kumar’s* SVM, which builds the probabilistic model identified above, does so by conducting data analysis and pattern recognition on one or more feature vectors.

4. *Claim 5 – The computer-implemented method according to claim 1, wherein the executable file is identified in response to a detected condition.*

135. *Kumar* discloses that the executable file is identified in response to a detected condition. This detected condition includes when a potential malware file is received at the customer front-end 112 or when it attempts to infect the customer machine front-end 112. For example, *Kumar* discloses that “[w]hen potential malware is received at the customer machine front-end 112, the potential malware is processed before being allowed to run, so as to detect whether the potential malware is actual malware 114. If the potential malware is actual malware 114, as in the example of FIG. 1, the malware 114 will attempt to infect the customer

machine front-end 112, as represented in FIG. 1 by circled numeral one (1).” *Kumar*, [0022]. Numeral (1) is shown below in Figure 1:



Kumar, Fig. 1; see also *id.*, Fig. 5 (“Receive unknown file”).

136. *Kumar* additionally discloses that the executable file is identified in response to another detected condition, which may be a probing attempt. Namely, *Kumar* discloses that “when probing is suspected, the code (instead of or in addition to any features and characteristic data) that was received may be analyzed in more detail. Even if malware is not detected by current classifiers, such a post-probing analysis may allow classifiers to be updated (e.g., at a high priority) based upon what is learned from the analysis. In other words, a malware author may be revealing new malware techniques during the probing attempts, and these can be detected and proactively used to protect against what may be future malware.” *Kumar*, [0050].

5. *Claim 6 – The computer-implemented method according to claim 5, wherein the detected condition is user request for a file download.*

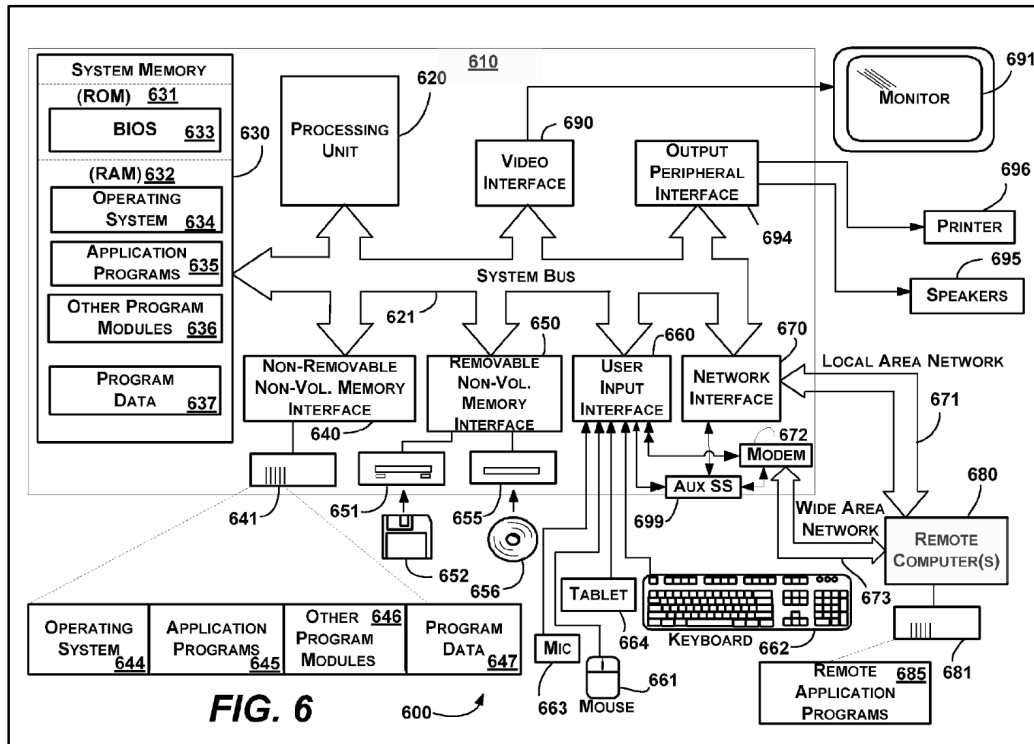
137. As discussed above in Claim 5, *Kumar* discloses that the executable file is identified in response to a detected condition, such as when the file is received, or when the file attempts to execute. In my opinion, it would have been obvious to a PHOSITA to identify the executable file in *Kumar*'s methods prior to receipt or execution, such as when the user sends a request for a file download. First, *Kumar* generally provides a process where the disclosed threat detection analysis is conducted before the file is downloaded. *See, e.g., Kumar*, [0036] (“the customer can be given the opportunity to wait for the backend service to upload and analyze the unknown file before downloading the file to the customer’s machine 112”). And separately, *Kumar* also discloses monitoring API call activity, which a PHOSITA would have understood to include requests for file downloaded. *Id.*, [0039]. Therefore, a PHOSITA would have found it obvious to modify *Kumar*'s methods to identify the file before downloading it upon a detected condition—the detected condition being the API call sent when requesting download. This modification would enhance *Kumar*'s threat detection analytics by identifying potential malware files earlier in the process—prior to downloading or execution.

138. A PHOSITA would have had a reasonable expectation of success making this modification as it requires only reprogramming well within the skill of

a PHOSITA to already present hardware. Indeed, a PHOSITA would have understood that the process for identifying a potentially malicious file to run malware detection processes on the file prior to download was known in the art at the time of the invention. For instance, U.S. Patent No. 8,875,289 B2 to Mahaffey et al. (“*Mahaffey*”) (Ex. 1028) states, “[f]or example, a user of mobile communication device 101 may wish to download a file to his device, but prior to installing the file, may wish to send the file or identifying data associated with the file to the server 151 in order to check if the file is malicious or otherwise undesirable.” *Mahaffey*, 7:46-50.

139. Alternatively, it would have been obvious to utilize a router or “proxy” in *Kumar*’s methods to detect malware before the file is downloaded to the client computer, such as upon request by the user to download the file. For example, *Kumar* discloses that a client computer 610 “may operate in a networked environment using

logical connections to one or more remote computers, such as a remote computer 680.” *Kumar*, [0081]. This is shown below in Figure 6:



Id., Fig. 6. *Kumar* explains that “remote computer 680 may be a personal computer, a server, a **router**...” *Id.*, [0081]. And this router “typically includes many or all of the elements described [] relative to the computer 610[,]” (*see id.*) which includes any of the elements described in paragraphs [0073]-[0080], including the program modules for malware detection described in *Kumar*’s disclosures. Based on *Kumar*’s disclosure, therefore, it would have been obvious for this router to detect a user request for download of a file, to intercept the file, and perform *Kumar*’s disclosed malware detection methods before downloading the file to the client computer 610.

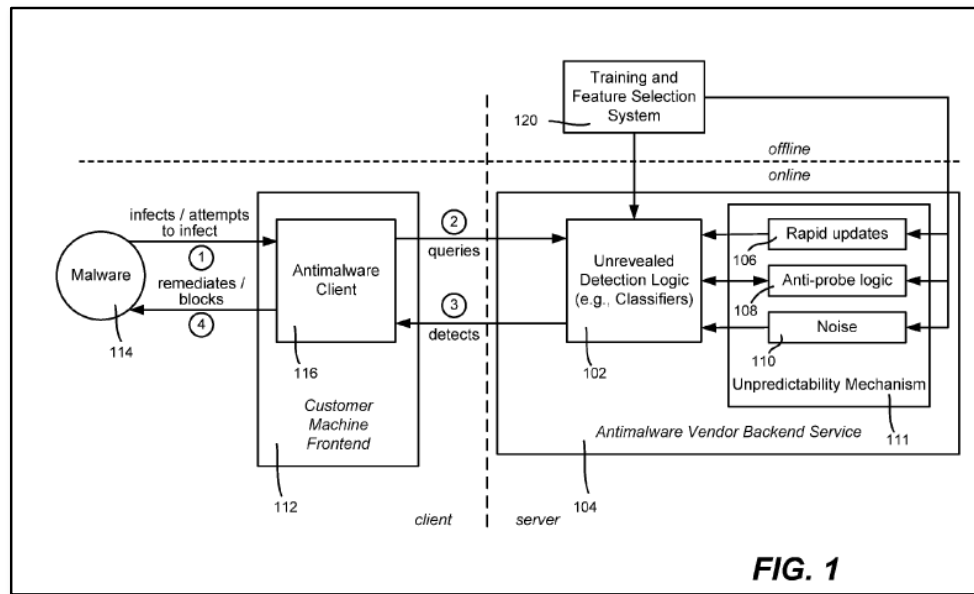
Doing so would offload computing tasks to the router gateway device, freeing up processing and user space on the client computer and significantly reducing the risks of running malicious files on the client computer.

140. A PHOSITA would have had success making this modification. As discussed above, *Kumar* already discloses performing its threat detection techniques in a networked environment. *See also id.*, [0075] (noting the “invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network”). Thus, this modification would have been well within the skill of a PHOSITA at the time of the invention.

6. *Claim 7 – The computer-implemented method according to claim 5, wherein the detected condition is the detection of a new file attempting to execute.*

141. As I described above, *Kumar* discloses that its system identifies a potentially malicious file when it “**attempt[s] to infect the customer machine front-end** 112, as represented in FIG. 1 by circled numeral one (1).” *Kumar*, [0022].

Figure 1 is reproduced below:



Kumar, Fig. 1. Based on Kumar’s description, a PHOSITA would have understood that as the potentially malicious file was attempting to infect the customer machine front-end, the file was “attempting to execute” on the customer machine front-end.

7. Claim 8 – The computer-implemented method according to claim 1, wherein the plurality of static data points represent predefined character strings in the executable file.

142. I understand that the ’869 Patent does not use the term “predefined character strings” outside the claims. However, the ’869 Patent generally provides exemplary static data points falling within the “Strings/Byte sequences” category. These exemplary static data points in the “Strings/Byte sequences” are shown below, boxed in red:

TABLE 1.1			
Numeric values	Nominal values	Strings/Byte sequences	Boolean values
File size	initialize	Comments	Address Of Entry Point Anomaly
linker version	Un-initialize	company name	Image Base Anomaly
code size	entry point	file description	Section Alignment Anomaly
OS version	subsystem	internal name	Size Of Code Mismatch Anomaly
image version	file subtype	legal copyright	Low Import Count Anomaly
subsystem version	language	original file	Entry Point Anomaly
file version number	file flags masks	private build	certificate Validity
product version number	file flags	product name	Certificate Exception
size of heap	file OS	special build	Code Characteristics Anomaly
size of stack	file type	product version	Code Name Anomaly
size of image	machine type	file version	Count Anomaly
PE header time	PE type	package code	Data Characteristics Anomaly
Section Entropy	section counts	product code	Data Name Anomaly
Sections count	DLL count	export DLL name	Export Exception
	DLL functions	assembly version	Large Number of DLLs Anomaly
	data directory	Certificate Issuer	flag DLL Name Anomaly
	export count	Certificate Subject	Number of Functions Anomaly
	Earliest Data Byte	Imports	Function Name Anomaly
	resources	Exports	PE Header Anomaly
	resources language	Section Names	High Section Count Anomaly
	resource Encoding	Non-resource section strings	PE Magic Validity
	resource code page		Resource Exception
	resource size		VR Code Ratio Anomaly
	DLL characteristics		Import Exception

'869 Patent, Table 1.1. The '869 Patent provides that this “string sequence data” may be “processed as n-grams[.]” *Id.*, 8:15-23.

143. I also understand that the PO has explained “‘character strings’ can be numerical values...used interchangeably with ‘byte sequences’[.]” Ex. 1009 (*Responsive Markman Brief, Dkt. 98*), 66 (citing parent '844 Patent, 8:29-48, “‘n-grams’ made up of ‘string and/or byte sequences’”). I have been asked to apply PO’s interpretation. I express no opinion as to the correctness of this interpretation.

144. Consistent with the teachings in the '869 Patent, *Kumar* discloses static data points such as “File Name,” “Certificate information” and “Version” that fall within the “Strings/Byte sequences” category the '869 Patent describes in Table 1.1. These static data points are reproduced below:

Feature	Description
File Name	Name of the PE file
Original File Name	File name in the original report
File Name Matches Original?	Whether the file name in this report matches the file name in the original report
File Type	Type of file it is
Signer Name	Organization that signed the file
Signing Authority	Certificate authority that issued the signature
Signature Type	File signed or not; if signed, whether the signature is legitimate or invalid
Description	Description of the file in the header
Organization	Manufacturer of the binary file
Version	Version number of the binary file
Packed	Whether the file is packed
LS Hash	Locality sensitive hash
Behavior Feature	Represents the simple behavior that caused the report to be generated

Kumar, [0028]. And like the '869 Patent, this data can be represented as 3-byte n-grams (trigrams). *Id.*, [0052]. It is my opinion, therefore, that *Kumar* discloses static data points as “*character strings*.”

145. It is also my opinion that these character strings are “predefined” as that term is understood based on the intrinsic record. For example, during prosecution of the parent '844 Patent, Applicant proposed several amendments clarifying that “static data points” correspond to “points of reference (in the file content)” such as to “particular sections or portions of the file.” '844 Parent FH (Ex. 1006), 134-135. I understand that the Examiner found that the prior art (Wojnowicz) disclosed “a string of hexadecimal characters” and that because an example static data point is “string information[,]” the prior art disclosed this concept. Applicant ultimately removed the “points of reference” language and added “*the plurality of static data*

points represent predefined character strings in the executable file[;]” suggesting that Wojnowicz’s “fixed-length file chunks do not represent predefined character strings” where “each fixed-length file chunk is, itself, a random (e.g. undefined) character string having no initial significance to the system of Wojnowicz.” Id., 312.

146. In the present challenge, *Kumar’s* filename, version, certificate information, etc. character strings correspond directly to particular predefined sections of the file. Namely, the file’s header. These strings thus are (pre)defined and would be highly significant to classification, in contrast to the file chunks in Wojnowicz’s disclosures.

8. *Claim 9 – The computer-implemented method according to claim 1, wherein a determination of whether the executable file is harmful is used to retrain the classifier.*

147. *Kumar* discloses its detection logic is “updateable” such that it “may be updated based upon one or more updating criteria, such as...new information and/or the like.” *Kumar*, [0006]. For example, *Kumar* discloses “**retraining** and/or replacing the logic[.]” *Kumar*, [0047]. A PHOSITA would have found it obvious for such “new information” to include a determination of whether the executable file is harmful to retrain the detection logic (which includes known classifiers as I explained in my opinions above). More specifically, it would have been obvious for *Kumar’s* training and feature selection system 120 to employ a well-known feedback approach to update its training dataset. In a feedback approach, the determination of

whether the executable file is malware (i.e., harmful) is fed back into the system as updated training data.

148. For example, there were several well-known feedback approaches at the time of the invention. First, U.S. Patent No. 9,349,103 to Eberhardt, III et al. (“*Eberhardt*”) (Ex. 1029) describes the application of machine learning classifiers to detect anomalies and malware in computing systems. *Eberhardt*, Abstract. To optimize training the machine learning classifiers, *Eberhardt* explains that generated “report[s] 204 may be fed back to the training system for future training and/or adjustment of the [machine learning] models 106.” *Id.*, 4:42-47; *see also id.*, Fig. 2 (showing the “Report 204” after the “Detection Module 107”). In this way, the system can “show...how similar an event is to a set of historic, adjudicated events that are, for example in a malware detection situation.” *Id.*, 5:34-45.

149. Second, U.S. Patent No. 9,043,894 to Dennison et al. (“*Dennison*”) (Ex. 1030) describes malicious software detection in a computing system which uses a suitable machine learning model for malware analysis. *Dennison*, Abstract. *Dennison* explains that data items scored as malware “can be fed back into the system to improve machine learning.” *Id.* Third, U.S. Patent No. 9,306,971 to Altman et al. (“*Altman*”) (Ex. 1031) describes a system and method for malware detection learning that implements adaptive machine learning algorithms. *Altman*, Abstract. *Altman* discloses that “[i]n some embodiments, malware detection results

of the system are fed back” such that this feedback can be used “for adapting the models and updating the reputation database or blacklist.” *Id.*, 3:48-61. Thus, it was well-known in the art for malware detection systems to implement a feedback approach to update a training dataset.

150. A PHOSITA would have had a reasonable expectation of success and have been motivated to facilitate retraining in *Kumar* using a well-known feedback approach, such as recursive feedback training where the determination of whether the executable file is malware (i.e., harmful) is fed back into the system in order to improve machine learning. Indeed, feeding back a malware determination would provide “signature data” to the detection logic, which *Kumar* explains is “needed to remediate the problem.” *Kumar*, [0024]-[0025].

9. *Claim 18(f) – evaluating the feature vector using a machine learning model to determine whether the executable file fits into one of the categories of files.*

151. I understand that claim 18 broadly claims a “machine learning model” as the component which evaluates the feature vector. In contrast, claim 1 more narrowly recites a “support vector machine” as the component which evaluates the feature vector. A PHOSITA would have understood that a support vector machine is a type of machine learning model as claimed in the ’869 Patent. *See, e.g.*, ’869 Patent at claim 21 (“**wherein the machine learning model comprises a support vector machine**”). Thus, for the same reasons that *Kumar*’s support vector machine

discloses claim 1, *Kumar's* support vector machine also discloses the machine learning model claimed in claim 18.

152. Additionally, *Kumar* discloses a list of other exemplary telemetry classifiers that may be used to assign an unknown telemetry report as malware or benign. For example, *Kumar* discloses the following list of telemetry classifiers compatible with its disclosed methods: “logistic regression, naïve Bayes, support vector machine, neural network, boosted decision tree, other boosted algorithms, ensemble classifiers, hierarchical classifiers and/or the like.” *Kumar*, [0027]; *see also id.*, [0039] (disclosing the same). A PHOSITA would have found it obvious to utilize any of these other types of classifiers—all of which are examples of machine learning models (or may implement machine learning models)—to accomplish *Kumar's* disclosed threat detection methods. Indeed, this would merely be the simple substitution of a known machine learning model for another to obtain predictable results. Such interchangeability would have been based on the desired classification parameters set by the user or system. A PHOSITA would have had success implementing any of its well-known classifiers, as *Kumar* envisions its system/methods performing with its disclosed example list of classifiers.

153. Moreover, it would be obvious to try any of *Kumar's* expressly stated classifiers in its system to predictably classify a file as malware or benign.

10. Claim 20 – The computer-implemented method according to claim 18, wherein the machine learning model comprises an artificial neural network.

154. As I explained in my opinions above, *Kumar* discloses a list of exemplarity telemetry classifiers which serve as a machine learning model, and this list includes a “neural network[.]” *Kumar*, [0027]; *see also id.*, [0039]. A PHOSITA would have understood that this “neural network” is an “artificial neural network.” Indeed, based on my experience with malware detection systems and machine learning models, an “artificial” neural network is often used synonymously or interchangeably with a “neural network.” All neural networks are “artificial” in that they rely on artificial intelligence, or machine learning, to perform the requisite processing. Consistent with my experience, U.S. Patent Publication No. 2007/0245520 to Yong et al. (“*Yong*”) (Ex. 1032) provides example “machine learning algorithms, such as ANN (Artificial neural Network), SVM (Support Vector Machines)” and others. *Yong*, [0035].

155. For the same reasons above in Claim 18(f), a PHOSITA would have been motivated to apply any of the listed classifiers, including a neural network, depending on the desired classification parameters. *See above*, Claim 18(f) (also noting this would have been obvious to try out of the list of choices to provide predictable results).

11. Claim 22 – The computer-implemented method according to claim 18, wherein the machine learning model comprises a machine learning decision tree.

156. *Kumar*'s list of exemplary telemetry classifiers which serve as a machine learning model also includes a “boosted decision tree[.]” *Kumar*, [0027]; *see also id.*, [0039]. A PHOSITA would have understood this “boosted decision tree” is a “machine learning decision tree” as claimed. Indeed, it was known in the art that boosted decision trees use machine learning for classification processing. *See, e.g.*, U.S. Patent No. 9,721,212 to Gupta et al. (“*Gupta*”) (Ex. 1033). *Gupta* relates malware protection using learning models. *Gupta*, Abstract, 9:26-29; *see also* U.S. Patent No. 9,324,034 to Gupta et al. (“*Gupta II*”) (Ex. 1034), Abstract (“applying machine learning techniques to generate a first family of classifier models using a boosted decision tree”). For the same reasons above in Claim 18(f), a PHOSITA would have been motivated to apply any of the listed classifiers, including the boosted decision tree, depending on the desired classification parameters. *See above*, Claim 18(f) (also noting this would have been obvious to try).

12. Claim 23 – The computer-implemented method according to claim 18, wherein the machine learning model comprises a Bayesian network.

157. *Kumar*'s list of exemplary telemetry classifiers which serve as a machine learning model also includes a “naïve Bayes” model. *Kumar*, [0027]; *see*

also id., [0039]. A PHOSITA would have understood the “naïve Bayes” model is a specific species of the “Bayesian network” genus. For example, U.S. Patent No. 10,783,254 to Sharma et al. (“*Sharma*”) (Ex. 1035) describes systems and methods for detecting malware in mobile applications. *Sharma*, Abstract. *Sharma* provides that its system can use a “Bayesian model classifier” “for example a Naïve Bayesian classifier or a Tree-Augmented Naïve Bayesian classifier.” *Id.*, 6:34-38; *see also id.*, 12:33-36 (“Variants of the Bayesian classifier can be used in various embodiments” such as the “Naive Bayesian classifier”). For the same reasons above in Claim 18(f), a PHOSITA would have been motivated to apply any of the listed classifiers, including the naïve Bayes model, depending on the desired classification parameters. *See above*, Claim 18(f) (also noting this would have been obvious to try).

13. *Claim 24 – The computer-implemented method according to claim 18, wherein evaluating the feature vector using the machine learning model comprises evaluating the feature vector using support vector processing.*

158. I understand that PO has stated the plain and ordinary meaning of “support vector processing” “is simply ‘processing that involves a support vector.’” Ex. 1009 (*Responsive Markman Brief*, Dkt. 98), 69 (noting these “can be processed both by support vector machines and by other machine learning classifiers, such as neural networks”). I have been asked to apply PO’s interpretation. I express no opinion as to the correctness of this interpretation.

159. As I described above, *Kumar* discloses machine learning models, such as “support vector machines” and “neural networks” used to classify a file. *See above*, Claim 1(f), 18(f), 20-23. Specifically, *Kumar* discloses:

In one implementation of the telemetry classifier, a multiclass classifier (e.g., logistic regression, naïve Bayes, support vector machine, neural network, boosted decision tree, other boosted algorithms, ensemble classifiers, hierarchical classifiers and/or the like) may be used to assign an unknown telemetry report to a particular malware family, with a generic malware family representing other families, a particular benign group of applications, and/or the generic benign class representing any remaining benign applications.

Kumar, [0027]. A PHOSITA would have understood that at least the “support vector machines” and “neural network” models include processing that involves a support vector and therefore perform “support vector processing” under PO’s interpretation of the claim language.

14. *Claim 25 – The computer-implemented method according to claim 18, wherein the machine learning model comprises a clustering model.*

160. As I discussed above, *Kumar* discloses a list of exemplarity telemetry classifiers that may be used to assign an unknown telemetry report to a particular malware family, the generic malware family, a particular benign family, or the generic benign family. This list of telemetry classifiers includes, “e.g., logistic regression, naïve Bayes, support vector machine, neural network, boosted decision tree, other boosted algorithms, ensemble classifiers, hierarchical classifiers and/or the like[.]” *Kumar*, [0027]; *see also id.*, [0039]. A PHOSITA would have understood

or found obvious that several of these models utilize “clustering.” For example, at least “ensemble classifiers” and “hierarchical classifiers” implement clustering techniques to classify data as malicious or benign.

161. Indeed, this was known in the art. “Ensemble Methods for Malware Diagnosis Based on One-class SVMs” by Xing An (“*Xing*”) (Ex. 1036) provides means to enhance typical SVMs with ensemble and hierarchical techniques. With respect to ensemble or hierarchical methods, *Xing* provides that “Bagging [] and clustering are two ensemble methods[.]” *Xing*, 2. In *Xing*’s experiments, *Xing* explains that “for clustering, we conducted a bottom-up hierarchical clustering along with the greedy approach for selecting the candidates in each iteration, the clustering process terminates when all the clusters have merged into one final cluster that contains all the training examples.” *Id.* Thus, *Xing* supports that ensemble and hierarchical methods utilize clustering techniques.

162. Moreover, it was known that support vector machines can cluster data. For example, “Malware behavioral detection and vaccine development by using a support vector model classifier” by Ping Wang et al. (“*Wang*”) (Ex. 1022) describes that:

Support vector machines (SVMs) [6,7] are used for clustering data into two categories according to maximum boundary geometry. Solving classification problems by using an SVM classifier (SVC) guarantees few classification errors caused by maximizing the generalisation ability of learning incorporating the Lagrange multiplier optimization algorithm [8].

Wang, 1013

163. Alternatively, to the extent PO or its expert argues that typical support vector machines do not cluster data, it would have been obvious to a PHOSITA to modify *Kumar's* support vector machine to implement well-known clustering techniques according to known benefits. For example, as discussed in *Wang* above, SVM clustering techniques “guarantees few classification errors caused by maximizing the generalization ability of learning[.]” *Wang*, 1013. A PHOSITA would have had a reasonable expectation of success modifying *Kumar's* SVM to perform clustering as was well-known in the art.

164. Additionally, it would have been obvious to employ other known clustering algorithms in *Kumar's* methods as the classifier. For example, *Tahan* provides known in the art clustering techniques. *See, e.g., Tahan*, 951 (describing known in the art “classification techniques” in the “context of benign/malware files classification” including “k-nearest neighbor (KNN)”). KNN is one of the most widely known clustering algorithms. *See, e.g., Tahan*, 951; *see also* U.S. Patent No. 9,306,966 to Eskin et al. (“*Eskin*”) (Ex. 1037), claim 21 (claiming process for determining a specific cluster as the k nearest neighbor to the selected data).

165. A PHOSITA would have been motivated and would have had success in employing a KNN clustering algorithm in *Kumar's* methods for the same reasons

identified above with implementing any known clustering algorithm in *Kumar's* methods.

15. *Claim 31 – The computer-implemented method according to claim 18, further comprising preventing execution of the executable file based on classifying the executable file as harmful.*

166. *Kumar* discloses when the potential malware is received at the customer machine front-end, “the potential malware is processed before being allowed to run, so as to detect whether the potential malware is actual malware 114.” *Kumar*, [0022]. In this regard, *Kumar* discloses that after malware is detected, the backend service “may provide the antimalware client component 116 with information (e.g., signature data) needed to remediate the problem. In this way, the antimalware client component 116 may take remedial action (corresponding to circled numeral four (4) in FIG. 1), such as to **clean delete or quarantine the file containing the malware 114.**” *Kumar*, [0025], Fig. 1 (circle (4) “remediates/blocks”). A PHOSITA would have understood that these remediation processes would prevent a files execution. Namely, cleaning, deleting, quarantining, or blocking a file would all prevent execution.

16. *Claim 46(e) – at least one processor configured to perform operation of:*

167. I understand that claim limitations 46(b) and 46(e) claim “at least one processor.” *See*, '869 Patent, Claim 46. To the extent PO or its expert argues that

these limitations require separate processors, a PHOSITA would have understood or found obvious that *Kumar* discloses multiple processors performing its claimed methods. For example, *Kumar* explains its methods are “performed at least in part **on at least one processor...**[,]” indicating that multiple processors may perform its disclosed methods. Moreover, *Kumar* expressly discloses that “multiple processing chips or multiple devices can share the performance of one or more functions described herein.” *Kumar*, [0085]; *see also id.*, [0073]-[0074]. Based on these disclosures, a PHOSITA would have understood that *Kumar* discloses multiple processors performing its described methods.

IV. CONCLUSION

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

Date: July 3, 2023



By:

Dr. Markus Jakobsson

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).

- **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; Kecker & Van Nest). Details and references upon request.
- **Publications.** Books: *Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft* (Wiley, 2006); *Crime-ware: 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

- [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

- [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.

- [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.

- [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

- [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. 157–173.
- [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.

- [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/](http://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-phishing logon authentication object oriented system and method, WO 2006062838 A1, 2005.
- [197] Jakobsson, Jean-Pierre Hubaux, Levent 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, Procédé et appareil permettant d'assurer la sécurité d'utilisateurs de dispositifs à capacités bluetooth, WO 2002019641 A2, 2001.
- [214] Robert M. Arlein, Ben Jai, Markus Jakobsson, Fabian Monroe, 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 Monroe, Michael Kendrick Reiter, Susanne Gudrun Wetzl, 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 Monroe, Michael Kendrick Reiter, Susanne Gudrun Wetzl, 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.