THE PARADOX OF SOURCE CODE SECRECY

Sonia K. Katyal†

† Haas Distinguished Chair and Chancellor's Professor of Law, University of California at Berkeley. Many, many thanks to the following for comments and conversation: Ken Bamberger, Ash Bhagwat, Andrew Bradt, Ari Chivukula, Ryan Calo, Christian Chessman, Danielle Citron, Jim Dempsey, Rochelle Dreyfuss, Charles Duan, Jeanne Fromer, Jim Gibson, Tait Graves, Ken Goldberg, Sue Glueck, James Grimmelmann, Zachary Hecht, Molly van Houweling, John Hamasaki, Gautam Hans, Chris Hoofnagle, Brett Kaufman, Neal Katyal, Logan Koepke, Joshua Kroll, Prasad Krishnamurthy, Amanda Levandowski, David Levine, Mark Lemley, Peter Menell, Robert Merges, Deirdre Mulligan, Tejas Narechania, Helen Nissenbaum, Claudia Polsky, Julia Powles, David Robinson, Andrea Roth, Bertrall Ross, Simone Ross, Pamela Samuelson, Sharon Sandeen, Andrew Selbst, Jason Schultz, Jonathan Simon, Vincent Southerland, Erik Stallman, Katherine Strandburg, Joseph Turow, Rebecca Wexler, Felix Wu, and John Yoo. A very sincere thanks to my colleague Bob Cooter, whose question at a workshop sparked this Article, and the organizers and participants of the Conference on Trade Secrets & Algorithmic Systems held at New York University Law School in November 2018. Joe Cera, Renata Barreto, Andrea Hall, and Aniket Kesari provided extraordinary research assistance.

INTRODUCTION

In October of 2017, before a jam-packed hearing at City Hall in New York, Council Member James Vacca listened to a series of testimonies that outlined two dramatically divergent visions for the future of technology, open data, and governance.[1] "This is the largest attendance a technology meeting has ever had," Vacca apparently said. "How am I going to top this next month?"[2] The occasion for the hearing was a bill with a lengthy (and seemingly snooze-worthy) title: "A Local Law to amend the administrative code of the city of New York, in relation to automated processing of data for the purposes of targeting services, penalties, or policing to persons."[3] Essentially, the bill required all agencies that use algorithms or other automated processing to publish their source code for public inves-

[1] Jessica McKenzie, *Hearing on Algorithmic Transparency Reveals Rift in NYC Tech Community*, CIVIC HALL (Oct. 19, 2017), https://civichall.org/civicist/hearing-algorithmic-transparency-reveals-rift-nyc-tech-community [https://perma.cc/S9XF-7T3G]; *see* Roshan Abraham, *New York City Passes Bill to Study Biases in Algorithms Used by the City*, MOTHERBOARD (Dec. 19, 2017, 9:52 AM), https://motherboard.vice.com/en_us/article/xw4xdw/new-york-city-algorithmic-bias-bill-law [https://perma.cc/6GLX-NTSG].

[2] McKenzie, *supra* note 1.

[3] Agenda, Hearing on Automated Decision Systems Used by Agencies, N.Y.C. COUNCIL, COMM. ON TECH. (Oct. 16, 2017, 1:00 PM), http://legistar.council.nyc.gov/MeetingDetail.aspx?ID=564867&GUID=9567478C-C9F4-4EDE-89F2-947E95A94ACD&Options=&Search [https://perma.cc/R2UE-QZ8K] (follow "Agenda" hyperlink).

tigation.[4] On a more abstract level, the bill would force the government to share its processes of automated government decision making and become essentially open source, reversing a long-standing trend toward opacity.[5]

For many who care about the future of democratic transparency, the proposal represented the culmination of their objective to situate the future of artificial intelligence (AI) within the parameters of democratic governance.[6] Almost immediately, however, the bill ignited a firestorm of debate that touched on the core of the underlying conflict between private property, the role of the government, and accountability. While nearly everyone applauded the impetus toward government transparency, some critics warned that increased disclosure would expose city systems to significant security risks, causing serious unintended consequences due to the proposal's breadth.[7]

Although concerns about government transparency are relatively straightforward, this Article argues that the issues raised by this debate underscore a growing divergence between the foundational tenets of intellectual property and its tension with AI. Ground zero for this conflict has become the murky,

---

[4] It also required agencies to provide outputs to the user. *See* Int. No. 1696–2017, N.Y.C. COUNCIL (Oct. 16, 2017, 1:00 PM), http://legistar.council.nyc.gov/MeetingDetail.aspx?ID=564867&GUID=9567478C-C9F4-4EDE-89F2-947E95A94ACD&Options=&Search [https://perma.cc/R2UE-QZ8K].

[5] McKenzie, *supra* note 1.

[6] *Id.*; *see also* Benjamin Herold, *'Open Algorithms' Bill Would Jolt New York City Schools, Public Agencies*, EDUC. WK. (Nov. 8, 2017, 12:43 PM), http://blogs.edweek.org/edweek/DigitalEducation/2017/11/open_algorithms_bill_schools.html [https://perma.cc/SFL9-3XCN] (noting the bill's potential impact on the use of educational algorithms).

[7] *See, e.g.*, Don Sunderland, Deputy Comm'r for Enter. and Sol. Architecture, Dep't of Info. Tech. and Telecomms., Testimony of the Department of Information Technology and Telecommunications on Int. 1696, A Local Law to Amend the Administrative Code of the City of New York, in Relation to Automated Processing of Data for the Purposes of Targeting Services, Penalties, or Policing to Persons (Oct. 16, 2017), https://www1.nyc.gov/assets/doitt/downloads/pdf/DoITT%20Testimony%20Int%201696%20FINAL.pdf [https://perma.cc/S6CR-LACH] (discussing before the Committee on Technology perceived flaws in the bill); *see also* Julia Powles, *New York City's Bold, Flawed Attempt to Make Algorithms Accountable*, NEW YORKER (Dec. 20, 2017), https://www.newyorker.com/tech/elements/new-york-citys-bold-flawed-attempt-to-make-algorithms-accountable [https://perma.cc/NVU8-AMWY] (acknowledging the potential harms that this legislation could have for contractual and proprietary interests). In the end, the Council passed a law creating a task force of experts to investigate New York City's use of algorithms, a move that represented a significant narrowing of the bill's original goals. *See* Devin Coldewey, *New York City Moves to Establish Algorithm-Monitoring Task Force*, TECHCRUNCH (Dec. 12, 2017), https://techcrunch.com/2017/12/12/new-york-city-moves-to-establish-algorithm-monitoring-task-force/ [https://perma.cc/22LV-V2VU].

messy intersection of software, trade secrecy, and public governance. Today, algorithms are pervasive throughout public law, employed in predictive policing analysis, family court delinquency proceedings, tax audits, parole decisions, DNA and forensic science techniques, and matters involving Medicaid, other government benefits, and educator evaluations.[8] And their results are often inscrutable, even though their results can demonstrate significant risk of bias.[9] In one example, ProPublica analyzed the recidivism risk scores of over 7,000 people arrested during a two-year period in Broward County, Florida, and found that only twenty percent of those predicted to commit future crime actually did so, and that the formula appeared to inaccurately flag black defendants as future criminals at twice the rate of white defendants.[10]

At their core, these automated systems often implicate central issues of due process, criminal (and civil) justice, and equal protection.[11] Yet, because their inner workings are often protected as trade secrets, they can remain entirely free from public scrutiny.[12] In all of these cases, for example, the source

---

8 *See* AI NOW INSTITUTE, LITIGATING ALGORITHMS: CHALLENGING GOVERNMENT USE OF ALGORITHMIC DECISION SYSTEMS 5 (2018), https://ainowinstitute.org/litigatingalgorithms.pdf [https://perma.cc/KZ52-PZAH] (noting these areas of use); *see also* Danielle Keats Citron, *Open Code Governance*, 2008 U. CHI. LEGAL F. 355, 356–57 (detailing government uses of automated decision making); A Local Law in Relation to Automated Decision Systems Used by Agencies Testimony, N.Y.C. COUNCIL, COMM. ON TECH. (Aug. 24, 2017) (Statement by Joshua North, Legal Aid Society), at 80–81, available at http://legistar.council.nyc.gov/LegislationDetail.aspx?ID=3137815&GUID=437A6A6D-62E1-47E2-9C42-461253F9C6D0&Options=ID%7cText%7cOther%7c&Search=1696 [https://perma.cc/4QLT-7X6M] (listing the ways algorithms are used in the criminal justice system for bail, predictive policing, DNA, family court, juvenile representation in delinquency proceedings, parole proceedings, and sex offender registration); AARON RIEKE, MIRANDA BOGEN & DAVID G. ROBINSON, PUBLIC SCRUTINY OF AUTOMATED DECISIONS: EARLY LESSONS AND EMERGING METHODS 3 (2018), https://www.omidyar.com/sites/default/files/file_archive/Public%20Scrutiny%20of%20Automated%20Decisions.pdf [https://perma.cc/H4DN-DXC4] (noting that the government uses algorithms to screen immigrants and allocate social services).

9 *See generally* Andrew D. Selbst & Solon Barocas, *The Intuitive Appeal of Explainable Machines*, 87 FORDHAM L. REV. 1085, 1087 (2018) (noting issues of opacity in decision making).

10 *See* Julia Angwin, Jeff Larson, Surya Mattu & Lauren Kirchner, *Machine Bias*, PROPUBLICA (May 23, 2016), https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing [https://perma.cc/EXU9-2JF9]. For a different perspective on the ProPublica study and related matters, see Arthur Rizer & Caleb Watney, *Artificial Intelligence Can Make Our Jail System More Efficient, Equitable and Just*, 23 TEX. REV. L. & POL. 181, 210–13 (2019).

11 *See* Rizer & Watney, *supra* note 10, at 197; *see also* N.Y.C COUNCIL, *supra* note 8, at 81 (Statement by Joshua North).

12 *See* Rebecca Wexler, *When a Computer Program Keeps You in Jail*, N.Y. TIMES (June 13, 2017), https://www.nytimes.com/2017/06/13/opinion/how-computers-are-harming-criminal-justice.html [https://perma.cc/G7GF-JGM4]

code that underlies and governs automated decision making is hidden from public view, comprising an unregulated "black box" that is privately owned and operated.[13]

This Article argues that the constitutionally inflected conflict that we now face is, in no small part, attributable to the failure of our system of intellectual property law to definitively address the boundaries of software protection and its implications for source code secrecy. As Pamela Samuelson recently put it, software protection has waxed and waned through copyright and patent protection at different points, at times extending the boundaries of protection, and at other times constricting it.[14] As a result, these uncertain and porous boundaries, subject to inconsistency, variation, and indeterminacy, have basically ushered in a system where the most risk-averse option, rationally, is to rely on trade secrecy to protect source code and to limit disclosure to the public as a result.

But this reliance on source code secrecy does not come without a price. Today, it appears that algorithms, rather than elected officials, are becoming a primary source of governance,

("The root of the problem is that automated criminal justice technologies are largely privately owned and sold for profit. The developers tend to view their technologies as trade secrets."). *See generally* Rebecca Wexler, *Life, Liberty, and Trade Secrets: Intellectual Property in the Criminal Justice System*, 70 STAN. L. REV. 1343 (2018) (discussing this problem) [hereinafter, Wexler, *Life, Liberty, and Trade Secrets*].

[13] *See generally* FRANK PASQUALE, THE BLACK BOX SOCIETY: THE SECRET ALGORITHMS THAT CONTROL MONEY AND INFORMATION (2015) (discussing this problem). For more on the issue of opacity in machine learning, see generally ROB KITCHIN, THE DATA REVOLUTION: BIG DATA, OPEN DATA, DATA INFRASTRUCTURES AND THEIR CONSEQUENCES (2014) (analyzing and summarizing the use of big data, open data, and data infastructures); Mike Ananny, *Toward an Ethics of Algorithms: Convening, Observation, Probability, and Timeliness*, 41 SCI. TECH. & HUM. VALUES 93 (2015) (discussing the ethical dilemmas in networked information algorithms); Jenna Burrell, *How the Machine 'Thinks': Understanding Opacity in Machine Learning Algorithms*, 3 BIG DATA & SOC'Y 1 (2016) (considering opacity in regards to the social consequences of algorithms related to personal and trace data); Danielle Keats Citron & Frank Pasquale, *The Scored Society: Due Process for Automated Predictions*, 89 WASH. L. REV. 1 (2014) (arguing for due process safeguards in the use of algorithms for those who are adversely impacted); Kate Crawford, *Can an Algorithm Be Agonistic? Ten Scenes from Life in Calculated Publics*, 41 SCI. TECH. & HUM. VALUES 77 (2016) (discussing the use of political theory to help understand how algorithms operate in public life); Nicholas Diakopoulos, *Algorithmic Accountability: Journalistic Investigation of Computational Power Structures*, 3 DIGITAL JOURNALISM 398 (2015) (considering how the hidden nature of algorithms reinforces societal power structures and biases); Tarleton Gillespie, *The Relevance of Algorithms*, *in* MEDIA TECHNOLOGIES: ESSAYS ON COMMUNICATION, MATERIALITY, AND SOCIETY 167 (Tarleton Gillespie et al. eds., 2014) (discussing how algorithms define and produce knowledge).

[14] Pamela Samuelson, *Staking the Boundaries of Software Copyrights in the Shadow of Patents*, 71 FLA. L. REV. (forthcoming 2019) (manuscript at 5), https://ssrn.com/abstract=3250496 [https://perma.cc/8TWY-Y8DM].

hidden from view.[15] Computer software appears in almost everything—computational biology, 3D printing, automobiles, home appliances, and much more.[16] But its dominance in the public sector of governance and AI, as I and others have argued, has become a significant source of concern in part due to the issue of privatization.[17] In a world of delegated decision making, the consistent power of closed code has a number of deleterious results for the public.[18]

This Article argues that source code carries a paradoxical character that is peculiar to software: the very substance of what is secluded often stems from the most public of origins, and often produces the most public of implications. It is the shortcomings of intellectual property law that have made this possible.

In this Article, I argue that the law of software has been willing to entertain a unique—and paradoxical—overlap between copyright, patent, and trade secrecy, even though the three regimes have somewhat opposing public goals. Copyright and patent law are oriented toward a spectrum that values dissemination and the circulation of ideas. In contrast, trade secrecy is motivated by opacity and seclusion. Yet software law has openly tolerated—indeed invited—a regime of opposites by

---

[15] For foundational perspectives on the view of code as governance, see generally LAWRENCE LESSIG, CODE AND OTHER LAWS OF CYBERSPACE (1999) (analyzing how cyberspace has changed regulation); Joel R. Reidenberg, *Lex Informatica: The Formulation of Information Policy Rules Through Technology*, 76 TEX. L. REV. 553 (1998) (suggesting that legislators need to understand information technology in order to regulate); James Grimmelmann, Note, *Regulation by Software*, 114 YALE L.J. 1719 (2005) (analyzing the impact of regulation on software); Kenneth A. Bamberger, *Technologies of Compliance: Risk and Regulation in a Digital Age*, 88 TEX. L. REV. 669 (2010) (discussing the power that regulation of software gives to computer programmers to determine compliance with minimal transparency). For an interesting, more recent account of the prospects of code regulation, see Paul Ohm & Blake Reid, *Regulating Software When Everything Has Software*, 84 GEO. WASH. L. REV. 1672 (2016).

[16] Manny Schecter, *The Changing Trade Secret and Patent Equilibrium*, TECHCRUNCH (June 20, 2016), https://techcrunch.com/2016/06/20/the-changing-trade-secret-and-patent-equilibrium/ [https://perma.cc/C9EL-HHQ9].

[17] *See generally* Sonia K. Katyal, *Private Accountability in the Age of Artificial Intelligence*, 66 UCLA L. REV. 54 (2019).

[18] Of course, it is important to note that there are many other areas of potential accountability and transparency aside from source code, including training data, data models, implementation guidelines, and even the business decisions that affect design and development. *See* WHITTAKER ET AL., AI NOW INST., AI NOW REPORT 2018, at II (2018), https://ainowinstitute.org/AI_Now_2018_Report.pdf [https://perma.cc/299L-92FQ] (listing these areas in addition to source code). While this Article primarily focuses on the intersection between trade secrecy and source code, these other areas (particularly the secrecy of training data) are important areas for future research as well. *See* Erik Stallman & Sonia Katyal, Contracting for Transparency (abstract on file with author).

enabling developers to commit to all three simultaneously, even though their underlying values can be at cross purposes. While this overlap of protection in software seems, at first glance, to be a good thing for the proprietary software industry, it has proven deleterious for the larger public in the context of automated decision making, particularly citizens who are now increasingly governed by an invisible hand that they can no longer investigate or question.[19] But, as I argue, this overlap may also be deleterious for other innovators as well.

Almost fifteen years ago, in a brilliant article, James Gibson identified the risks to democracy that inhere in closed code, particularly regarding its potential to encroach upon our everyday lives without transparency or accountability.[20] Those fears are no longer speculative; they have become an everyday reality for criminal defendants and others who are swept up by the specter of automated government decision making.[21] As a result, it is entirely possible to imagine a world where all of us face some form of automated regulation—all without detection, in part because the code is closed from public view and investigation.[22]

While many software scholars have focused on issues regarding copyright and patent protection, I argue that a greater focus on trade secrecy—and specifically source code secrecy—is gravely overdue in these current circumstances.[23] In this Article, I investigate an overlooked paradigm associated with source code, one that stems from the current failures of both private and public law to incentivize disclosure, leading to a domain where source code is largely dominated by trade secrecy. In both abstract and practical terms, this failure to in-

[19] *See generally* VIRGINIA EUBANKS, AUTOMATING INEQUALITY: HOW HIGH-TECH TOOLS PROFILE, POLICE, AND PUNISH THE POOR (2017) (arguing that government use of automated data further disenfranchises the poor); SAFIYA UMOJA NOBLE, ALGORITHMS OF OPPRESSION: HOW SEARCH ENGINES REINFORCE RACISM (2018) (discussing the way algorithms in search engines perpetuate oppression and create new kinds of racial profiling); CATHY O'NEIL, WEAPONS OF MATH DESTRUCTION (2016) (noting that although algorithms are seen as neutral because of their mathematical basis, they perpetuate discrimination).

[20] James Gibson, *Once and Future Copyright*, 81 NOTRE DAME L. REV. 167, 190 (2005).

[21] N.Y.C. COUNCIL, *supra* note 8, at 80–81 (Statement by Joshua North).

[22] *See* David Lyon, *Surveillance as Social Sorting: Computer Codes and Mobile Bodies*, *in* SURVEILLANCE AS SOCIAL SORTING: PRIVACY, RISK, AND DIGITAL DISCRIMINATION 13, 13 (David Lyon ed., 2003), http://www.felfel.is/sites/default/files/2016/Lyon,_D._(2003)._Surveillance_and_social_sorting%26_computer_codes_and_mobile_bodies%20(1).pdf [https://perma.cc/G7RQ-XKWJ].

[23] It bears mentioning that this Article is written mostly for a non-tech expert audience. For a related and excellent study of the role of trade secrecy in criminal proceedings, see Wexler, *Life, Liberty, and Trade Secrets*, *supra* note 12.

centivize disclosure has produced significant public law implications, ones that we are now grappling with due to the rise of AI.

To understand further the origins of source code secrecy and its implications, however, we need to look back through the complicated history of legal protection for software. In the first half of this Article, focusing primarily on intellectual property law, I describe the dominance of trade secrecy over source code, attributing it to a complex, dyadic relationship between law and the marketplace over the last several decades. As I describe, the specific qualities of software, with its short shelf life and abstract qualities, seem at first glance to be an imperfect fit for patent and copyright protection. Yet both areas of law were extended to protect software after some period of reluctance, leading to a regime where these different areas of law were essentially treated as complementary to trade secret protection.

Despite the extension of copyright and patent protection over software (or perhaps because of it), software garnered a unique position within the law: it remains one of the few spheres to enjoy concurrent protections from trade secrecy, copyright law, and patent law. Yet this state of affairs has produced dramatic implications for both the surrounding software industry and the public, who have become increasingly dependent on mass market software.

In the second half of the Article, I examine the implications of this shift toward mass market software for the public interest. Here, I examine the increasing rise of "closed code governance," which involves government's delegation of core government functions to private, automated decision making.[24] As I show, the consequences of this reliance on automation are particularly significant for marginalized groups who are often governed by closed code without a formidable ability to challenge or address their situation due to trade secrecy.

This Article has five parts. In Parts I and II, after a brief introduction to software and source code, I outline how both administrative and common law decisions have invited the coexistence of copyright and trade secrecy, allowing software to be widely disseminated and yet consistently underscored by source code secrecy at the same time.[25] In Part III, turning to software patentability, I argue that the shifting boundaries of

24 *See* Citron, *supra* note 8, at 360.

25 Schecter, *supra* note 16, at 190.

protection have produced a more complicated story. Since the boundaries of software patentability have also narrowed, trade secrecy becomes an even more attractive default avenue for protection, essentially displacing all other possibilities.

In Part IV, turning toward public law, I examine the civic implications of source code secrecy. Here, I argue that closed source code produces a dilemma for public transparency in an age of AI. At a time when so many government functions are being delegated to private companies, the rise of trade secrecy raises critical questions of accountability and oversight. In the final part of this Article, I address how governments—and courts—can address this problem, both through common law and regulatory reform.

In Part V, I make a case for limiting source code secrecy in certain contexts, offering an architecture of what I call "controlled disclosure." The Article concludes with a brief discussion of ways to offer greater transparency for source code and automated decisionmaking through reforming areas of intellectual property, contract law, and discovery. Here, I argue that the particular significance of source code necessitates a more granular set of efforts by legislators and courts toward transparency. Finally, returning to the City Council law that opened this paper, I offer a modest set of possibilities to engage greater norms toward disclosure in cases of significant public interest.

I
SOURCE CODE SECRECY AND COPYRIGHT

Over twenty years ago, Lawrence Lessig famously proclaimed, "Code is law."[26] That statement, at the time, was largely taken to suggest that computer code regulated human conduct in the same way that law regulated human conduct.[27] Today, however, many years later, we see that Lessig's observation was more than just a metaphor for regulating human behavior. In our modern age of algorithms, it is literally the case that code *is* law, and that law is code, because our government has delegated so many of its functions to automated decision making.[28] Yet to understand both the rise of trade secrecy, and its significant implications for democratic transparency, we must start with studying the history of code, its emergence, and its relationship to other areas of intellectual property.

26 LESSIG, *supra* note 15, at 5.

27 *Id.*

28 *See* Citron, *supra* note 8, at 360; Ohm & Reid, *supra* note 15, at 1673 (noting how physical functionality of devices has become replaced by code).

Since the onset of the computer age, the law has struggled to find a way to protect software through intellectual property principles, and it has rarely reached a consistent conclusion. At various times, different types of legal protection—copyright, patent, trade secret—have all dominated the landscape, leaving software law to become an area of considerable murkiness. In an influential article, now-Justice Stephen Breyer expressed concerns about the harm that might result from copyrighting software, reasoning that copyrighting code would increase transaction costs and impede the sharing of information that characterized the industry's expansion.[29] If parties had to license content from others, Breyer argued that they would expend efforts on designing around protected code, wasting precious resources to avoid litigation.[30]

As Breyer's observations suggest, software's integration with hardware, coupled with the absence of protections in copyright and patent law, led to an initial focus on trade secrecy and contract law for protection, what some have described as the first phase of software protection under intellectual property law.[31] Later, as mass market licenses entered the picture, leading to greater separation between hardware and software, copyright law became an increasingly attractive engine for protection, facilitated in no small part by a cadre of commentators and Congressional leaders who urged greater propertization, ushering in a second phase of protection.[32]

Afterward, in the early 2000s, software entered yet another shift, one that has been described as a third phase of protection, attributable to the limited scope of copyright protection and the increasing attractiveness of software patenting.[33] At first, this third phase seemed to offer developers some certainty of protection by enabling parties to pursue patentability in addition to the other options.[34] Copyright law protected software;

29 Stephen Breyer, *The Uneasy Case for Copyright: A Study of Copyright in Books, Photocopies, and Computer Programs*, 84 HARV. L. REV. 281, 348 (1970).

30 *Id.*

31 *See* Bradford L. Smith & Susan O. Mann, *Innovation and Intellectual Property Protection in the Software Industry: An Emerging Role for Patents?*, 71 U. CHI. L. REV. 241, 242 (2004) (describing various phases of the software industry's development, starting with contract law).

32 *See id.* at 242, 245.

33 *Id.* at 242.

34 MARK A. LEMLEY, PETER S. MENELL, ROBERT P. MERGES & PAMELA SAMUELSON, SOFTWARE AND INTERNET LAW 3 (3d ed. 2006).

patent law protected computer hardware and, increasingly, new processes and structures embodied in software.[35]

Today, due in no small part to the narrowing of software patentability and other forces, I would argue that we see that software's relationship to intellectual property law is now engaged in yet another revision. This fourth phase (if it can even be described as such) demonstrates a robust reliance on the backdrop of trade secrecy at the cost of more disclosure-oriented regimes like copyright and patent law. As I show in the Parts below, copyright and patent developments in software did little to incentivize disclosure, making trade secrecy even more attractive as a default mode of protection.

A. Code: An Introductory (and Incomplete) History

On a very basic level, a computer can perform a variety of different functions depending on the software it is fed.[36] These instructions to the computer are comprised of binary digits—ones and zeroes—and encode, step by step, a series of directions to the computer's physical hardware. This chain of ones and zeroes is called a computer's "object code" and is largely unreadable by humans.[37]

The first programming languages were originally motivated by the desire to replace the painstaking nature of specialized code with mathematical formulas.[38] Fortran, the first widely known computer language, was introduced by IBM in 1957.[39] Eventually, programmers began to develop other kinds of computer languages, like BASIC, Pascal, and C.[40] These high-level

---

[35] *Id.* at 3 (concluding in 2006 that "the main contours of legal protection for computer technology are relatively clear").

[36] Gibson, *supra* note 20, at 174.

[37] *Id.*

[38] Niklaus Wirth, *A Brief History of Software Engineering*, IEEE ANNALS HIST. COMPUTING, July-Sept. 2008, at 32–33.

[39] *Id.*

[40] Gibson, *supra* note 20, at 174. This summary of software history is admittedly all too brief. For various perspectives on the history of computing, see generally Thomas Haigh, *Historical Reflections: The Tears of Donald Knuth*, 58 COMMS. ACM 40 (2015); Martin Campbell-Kelly, *The History of the History of Software*, IEEE ANNALS HIST. COMPUTING, Oct.-Dec. 2007, at 40. Donald E. Knuth & Luis Trabb Pardo, *The Early Development of Programming Languages*, *in* A HISTORY OF COMPUTING IN THE TWENTIETH CENTURY 197 (N. Metropolis et al. eds., 1980); STEVE LOHR, GO TO: THE STORY OF THE MATH MAJORS, BRIDGE PLAYERS, ENGINEERS, CHESS WIZARDS, MAVERICK SCIENTISTS AND INCONOCLASTS—THE PROGRAMMERS WHO CREATED THE SOFTWARE REVOLUTION (2001); GLYN MOODY, REBEL CODE: INSIDE LINUX AND THE OPEN SOURCE REVOLUTION (2001); James W. Cortada, *Researching the History of Software from the 1960s*, IEEE ANNALS HIST. COMPUTING, Jan.–Mar. 2002, at 73; JEAN E. SAMMET, PROGRAMMING LANGUAGES: HISTORY AND FUNDAMENTALS (1969); HISTORY OF PROGRAMMING LANGUAGES (Richard L. Wexelblat ed., 1981);

languages, while still largely intelligible to only the most skilled programmers, came to be known as "source code," in part because they abstract away from the object code.[41] While the definition of object code seems relatively straightforward, source code can be defined in both broad and narrow terms.[42] But it essentially comprises everything that matters in software. Source code represents the commands that control a computer program, comprising a series of alphanumeric characters that are legible to humans.[43] Since computers only understand object code, use of a compiler is necessary to translate the source code into assembly code, which is an intermediate-level language; an assembler then translates the assembly code into object code.[44]

But source code is much more than just lines of commands—it comprises the lifeblood of software, embodying both the potential of the creativity that produces the code and the functionality that the code achieves. Although it mainly generates ready-to-use binaries, source code is essential for a variety of other practical reasons.[45] From a developer's perspective, it is generally considered much more versatile and informative

---

Thomas Ball, A Brief History of Software—From Bell Labs to Microsoft Research, 2009 6th IEEE International Working Conference Mining Software Repositories (May 16, 2009), *in* IEEE XPLORE, June 2009. Christof Ebert, *A Brief History of Software Technology*, IEEE SOFTWARE, Nov.–Dec. 2008, at 22.

41 Gibson, *supra* note 20, at 174.

42 A typical description of source code in litigation is the following:

> source code, object code (i.e., computer instructions and data definitions expressed in a form suitable for input to an assembler, compiler, or other translator), any text written in any high-level programming language defining firmware and/or software functionalities implemented on an integrated circuit, microcode, register transfer language ("RTL"), firmware, and hardware description language ("HDL"), as well as any and all notes, annotations, and other comments of any type related thereto and accompanying the code. For avoidance of doubt, this includes source files, make files, intermediate output files, executable files, header files, resource files, library files, module definition files, map files, object files, linker files, browse info files, and debug files.

David Maiorana, *Diagrams Not Considered Source Code Under Modified Protective Order*, JONES DAY (Nov. 10, 2017), http://jonesdayitcblog.com/source-code-modified-protective-order/[https://perma.cc/VSV3-3M6B].

43 Christian Chessman, Note, *A "Source" of Error: Computer Code, Criminal Defendants, and the Constitution*, 105 CALIF. L. REV. 179, 181 (2017).

44 Gibson, *supra* note 20, at 175; *see also* Glenn J. MacGrady, *Protection of Computer Software—An Update and Practical Synthesis*, 20 HOUS. L. REV. 1033, 1036 (1983) (explaining the conversion of source code into machine-readable and loadable instructions). Note, however, that many computer languages today, Javascript being one example, are not compiled to object code but are interpreted instead.

45 *See Source Code Definition*, LINUX INFO. PROJECT, http://www.linfo.org/source_code.html [https://perma.cc/UAP6-PR22] (last updated Feb. 14, 2006).

than object code, since access to the source code usually ensures that the system administrator can better tailor the software to particular requirements.[46] Having access to the source code also means that it is easier to fix bugs, determine error rates, respond to viruses, or locate other forms of malicious content.[47] It is also a core source of information to ensure interoperability, enhances learning for both new and experienced programmers, and assists with the purposes of software reusability.[48]

Yet because source code and software are often synonymized and treated alike in the case law and literature, it is often hard to realize, on a more granular level, that much of the case law involving source code involves something that is generally secret. The public prominence of software often overshadows its private, secret source code. However, the best way to figure out how a program actually works, particularly to assess its reliability and accuracy, is to start by reading the source code.[49]

B. The Birth of Source Code Secrecy

If we are to understand the rise of secrecy in software, then we must start at the place where mass market software began. One of the biggest shifts in computing took place in the late 1950s when computers, which had previously only been available to research institutions and universities, began to enter the world of business.[50] Initially, many companies developed software in house to keep up with the demands of customization.[51] This meant that most agreements were governed by contract law and, relatedly, trade secrecy, rather than other forms of intellectual property protection. Yet, over time, software development firms began to recognize that more and more clients were demanding the same sorts of projects, and they began to develop programs for a wider market.[52] As computing capacity began to expand, more attention came to be paid to the value of automation and structured programming.[53]

Around this time, the field of computer science began to emerge, largely out of the recognition that programming lan-

46 *Id.*
47 *Id.*
48 *Id.*
49 Chessman, *supra* note 43, at 182.
50 Wirth, *supra* note 38, at 32.
51 LEMLEY ET AL., *supra* note 34, at 3.
52 *Id.*
53 Wirth, *supra* note 38, at 33.

guages did not fit either the domain of mathematics nor electronics.[54] In 1975, software developers showed that high level languages could be used on microcomputers, reducing the need for expensive, sophisticated compilers.[55] Around this time, more and more software firms began to emerge to satisfy the more general purpose needs of their customers.[56] Thus, the market for software began to expand from custom programming to the development of products that required very little customization.[57] At that point, as expert Niklaus Wirth describes, "[s]uddenly, there was a mass market. Computing went mainstream."[58]

The computer industry grew by leaps and bounds from the 1960s to the 1970s, so that by the end of the 1970s, almost one hundred percent of Fortune 500 companies used computers.[59] By the end of the 1970s, almost fifty percent (or more) of the software used by organizations consisted of commercially available packages.[60] Although most developers had been relying on simple contract law (coupled with confidentiality provisions) to govern disputes, given the increased mass market potential, the industry turned to copyright law to seek protection.[61] But their efforts became complicated by the increasing complexity of the process of software development. In the 1960s and 1970s, for example, the industry began to actively differentiate the designing of software from the development of code; computer scientists focused on design principles first and then on writing computer code second.[62]

As software became more complex, the role of the software engineer started to look less and less like a traditional "author" of the code.[63] The advent of software engineering dramatically increased the complexity of programs, bringing both modularization and structure, but it also contributed to a growing division between what came to be known as "literal" versus

54 *Id.*

55 *Id.*

56 LEMLEY ET AL., *supra* note 34, at 31.

57 *Id.*

58 Wirth, *supra* note 38, at 35.

59 Cortada, *supra* note 40, at 73.

60 *Id.*

61 LEMLEY ET AL., *supra* note 34, at 32.

62 *See* Wirth, *supra* note 38, at 32–33; *see also* Michael S. Mahoney, *What Makes the History of Software Hard*, 30 IEEE ANNALS HIST. COMPUTING, July–Sept. 2008, at 8 (describing the emergence of software engineering).

63 *See* Wirth, *supra* note 38, at 34–35.

"nonliteral" forms of protection.[64] By diverging from the literal, code-based characteristics of software from the previous era, these programs opened the door to more challenges under copyright protection because they toed a fine line between idea and its expression, and thus were vulnerable to merger-related challenges.[65]

What emerges, then, from this (admittedly brief and incomplete history) is that software increasingly became more than just a program, it began to comprise also the design, involving more abstract ideas, rather than just code.[66] As software systems grew in complexity, the concept of modularization began to take on greater significance, and the rise of the personal workstation led, in no small part, to the development of the concept of object orientation, which led to the creation of windows, buttons, toolbars, icons, and menus.[67] By the mid-1980s, enormous advances in hardware led to a massive rise in computing power, blending the fields of computer and communications technologies with the advent of the Internet.[68]

During the last decade, of course, perhaps the most attention has been focused on the development of AI, which is a field that develops computer systems to perform tasks normally performed by humans, including those that implicate learning and decision making.[69] AI has grown significantly in recent years, in no small part due to the development of machine learning, which relies on developing algorithms that can create analytical models from data, without relying on a human to program a solution.[70] Before the advent of machine learning, software developers had to manually code a variety of functions into a system; today, machine learning can do all of this much more

[64] At the lowest level of abstraction is the source or object code of a computer program, its literal element. A higher level of abstraction involves things like design features (its "architecture"), which constitute nonliteral elements. *See* LEMLEY ET AL., *supra* note 34, at 35 (noting this distinction).

[65] *See id.*

[66] *See id.*

[67] Wirth, *supra* note 38, at 37.

[68] *Id.* In the last few decades, computer-aided software engineering (CASE) provided automated assistance in software design and development.

[69] *Digital Decision-Making: The Building Blocks of Machine Learning and Artificial Intelligence*, 115th Cong. 2 (2017) (statement of Dario Gil, Vice President, AI and Quantum Computing, IBM).

[70] *Id.* at 2. For an explanation, see Nizan Geslevich Packin & Yafit Lev-Aretz, *Learning Algorithms and Discrimination*, *in* RESEARCH HANDBOOK ON THE LAW OF ARTIFICIAL INTELLIGENCE 88 (Woodrow Barfield & Ugo Pagallo eds., 2018) (noting that "machine learning is nonparametric and does not involve devising any particular mathematical model in advance").

efficiently.[71] In addition, advances in processing speed and power, and the emergence of specialized processing devices like graphical processing units, have enabled the use of artificial neural networks in a variety of embedded technologies and home devices.[72]

All of these developments, while great for the software industry, have posed complexities for intellectual property law, which has maintained relatively porous boundaries around areas of software protection. These shifts also usher in a kind of inescapable hybridity between literal and nonliteral forms of software protection.[73] As one commentator explains:

> [S]oftware is a very cumbersome expression of an idea. If asked about details of a software system by a mid-level manager, a programmer would never hand that manager pages of computer code, but instead, would choose an intermediate level of the design, perhaps a combination of some dataflow diagrams and some text description, to express her idea. . . . The design expresses the idea and the code expresses the idea; in the modern software engineering environment, the two are inextricably tied. The design represents the code and, as demonstrated above, the design is the code.[74]

As a result, software in and of itself is a chimera: it can be classified so narrowly that it can fall into multiple categories of intellectual property protection; or, it can be classified so broadly that it fits into none of them at all. And the law has supported this variance with its own shifting boundaries of intellectual property protection.

C. The Copyrightability of Software

In the early years of software development, particularly from the 1960s to the 1980s, programmers regularly shared source code, in part because much of the core aspects of computer operating systems were developed in an academic setting or in central corporate research labs with a great deal of auton-

[71] *Digital Decision-Making: The Building Blocks of Machine Learning and Artificial Intelligence*, *supra* note 69, at 2 (statement of Dario Gil, Vice President, AI and Quantum Computing, IBM).

[72] *Id.*

[73] Joseph G. Arsenault, *Software Without Source Code: Can Softwawre Produced by a Computer Aided Software Engineering Tool Be Protected?*, 5 ALB. L.J. SCI. & TECH. 131, 143 (1994) (questioning whether the software design is copyrightable, and if so, at what level it is protectable).

[74] *Id.* at 156.

omy.[75] In these settings, highly cooperative software development projects emerged, with little effort made to establish the boundaries of intellectual property ownership or to restrict reuse.[76]

Soon after the introduction of high-level programming languages like FORTRAN and others, software developers began to turn to contract law, along with copyright, patent, and trade secret law to protect their work.[77] Early programmers wrote software much like authors wrote manuscripts: they would come up with an idea and write down the program necessary to make the idea come to fruition.[78] A program, therefore, comprised a sequence that ran from the beginning to its end, and the programmer would write and rewrite the code until it accomplished its task.[79] In such cases, protection against verbatim copying was usually enough to protect the information.[80]

Although the original Copyright Act understandably made no reference to computer programs,[81] the Copyright Office in the mid-1960s began to allow registration—concluding that computer programs were readable, written works of authorship, but noting that the registrations could only issue under its "rule of doubt."[82] Yet this move represented a first bold step toward hybridizing copyright and trade secret protection in mass market software. As Diane Zimmerman explains,

> [t]his [mass-market] change led those in the software industry to see the advantage in trying to take advantage of copyright while retaining the benefits of trade secrecy. The use of copyright would enable them to distribute copies of their works in object code (that is, computer-readable) form to the public backed up by the threat of sanctions for infringement . . . . At the same time, developers wanted to maintain the economic value of their programs and ward off competi-

75 Josh Lerner & Jean Tirole, *The Simple Economics of Open Source* 200 (Nat'l Bureau of Econ. Research, Working Paper 7600, 2000), http://www.nber.org/papers/w7600.pdf [https://perma.cc/6ES3-LB5R].

76 *Id.*

77 Gibson, *supra* note 20, at 176.

78 Arsenault, *supra* note 73, at 142.

79 *Id.* at 144.

80 *Id.* at 149.

81 Richard Raysman, *Protection of Proprietary Software in the Computer Industry: Trade Secrets as an Effective Method*, 18 JURIMETRICS J. 335, 337–38 (1978).

82 Deposit of Computer Programs and Other Works Containing Trade Secrets, 48 Fed. Reg. 22,897, 22, 951 (May 23, 1983) (to be codified at C.F.R. pt. 202); Jay Dratler, Jr., *Trade Secret Law: An Impediment to Trade in Computer Software*, 1 SANTA CLARA COMPUTER & HIGH-TECH. L.J. 27, 42, n.64 (1985) (observing that object code was protected almost entirely by copyright law until the early 1980s).

tion by keeping the expression that embodied the design of these programs—their source code—a secret.[83]

Both objectives, Zimmerman writes, were achieved by convincing Congress to adopt a rule of doubt,[84] suggesting that the Copyright Office deferred to the courts' judgment.[85] Later, Congress established a National Commission on New Technological Uses of Copyrighted Works (CONTU), which concluded that copyright was the most appropriate form of protection for computer programs.[86] As Peter Menell has explained, at the time that CONTU was created, neither patent nor copyright had played a key role yet because the industry had developed mostly in reliance on trade secret protection and contract law instead.[87]

In these early days, computers were so specialized that they were not sold through traditional retail channels, and since hardware and software were often bundled together, there was only a minimal need to consider separate protection for software.[88] For hardware, patent protection ensured an adequate reward for the cost of innovation.[89] Thus, at least initially, contract law and trade secrecy provided much of the necessary protection against misappropriation, leading one leading commentator to conclude in 1978 that "[t]rade secret protection is, without question, the most effective current means of protecting valuable computer software," noting that one of the greatest drawbacks to patent and copyright was the requirement of disclosure.[90] During this period, companies relied heavily on secrecy and contract law; for example, in 1983, IBM started to include restrictions on the distribution of its source code for its operating systems, and also to require licensees to agree to refrain from reverse engineering.[91] In the years afterward, many more companies followed suit. But as the

[83] Diane Leenheer Zimmerman, *Trade Secrets and the 'Philosophy' of Copyright: A Case of Culture Clash*, *in* THE LAW AND THEORY OF TRADE SECRECY: A HANDBOOK OF CONTEMPORARY RESEARCH 299, 301 (Rochelle C. Dreyfuss & Katherine J. Strandburg eds., 2011).

[84] *Id.*

[85] LEMLEY ET AL., *supra* note 34, at 35.

[86] Samuelson, *supra* note 14, at 11.

[87] Peter S. Menell, *The Challenges of Reforming Intellectual Property Protection for Computer Software*, 94 COLUM. L. REV. 2644, 2652 (1994).

[88] LEMLEY ET AL., *supra* note 34, at 33. For an interesting discussion of the source of the distinction between hardware and software, see James Grimmelmann, *The Structure and Legal Interpretation of Computer Programs* 18 (2019) (draft on file with author).

[89] LEMLEY ET AL., *supra* note 34, at 33.

[90] Raysman, *supra* note 81, at 350.

[91] Dratler, *supra* note 82, at n.64.

mass market for software began to develop, it became clearer and clearer that developers needed other forms of protection as well.[92]

1. *Early Accommodations of Trade Secrecy*

Throughout the history of intellectual property's relationship with software, concerns about the secrecy of source code have carried a special significance given the potential overlap between trade secrecy and copyright. Initially, the Copyright Office required deposit of the full source code, just as it did for every other copyrighted work. Yet this proved to be a powerful initial deterrent to copyrightability, as trade secret law had already been the default mechanism.[93] Because of the fear of disclosure, only about 1,200 copyright registrations were issued between 1966 and 1978.[94] During this period, most of the registered programs belonged to the largest computer hardware manufacturers, who were in a better position to copyright programs and to disclose the nature of the programs to the public, because they stood to make more profit from selling hardware than software.[95]

Nevertheless, the choice to extend copyright protection to software, at that point, seemed like a speculative gamble in order to protect a nascent field of technology.[96] As the leading casebook on the topic explains:

> As CONTU recognized, it was impossible in 1978 to establish a precise line between copyrightable expression of computer programs and the uncopyrightable processes that they implement. Yet the location of this line—the idea/expression dichotomy—was critical to the rough cost-benefit analysis that guided CONTU's recommendation. Drawing the line too liberally in favor of copyright protection would bestow strong monopolies upon those who develop operating systems that become industry standards and would thereby inhibit other creators from developing improved programs and computer systems. Drawing the line too conservatively would allow programmers' efforts to be copied easily, thus discouraging the creation of all but modest incremental advances.[97]

---

[92] LEMLEY ET AL., *supra* note 34, at 4.

[93] *Id.* at 34.

[94] *Id.*

[95] Raysman, *supra* note 81, at 338.

[96] LEMLEY ET AL., *supra* note 34, at 33; *see* Note, *Copyright Protection of Computer Program Object Code*, 96 HARV. L. REV. 1723, 1724 (1983) (recommending protection).

[97] LEMLEY ET AL., *supra* note 34, at 35; *see also* Peter S. Menell, *An Epitaph for Traditional Copyright Protection of Network Features of Computer Software*, 43

As a consequence of reviewing the results of the first few years of software protection, in 1989 Congress decided to facilitate a remarkable break from its previous system: it decided to forego the deposit requirement for source code and set up a new system to respect the secrecy of source code instead.

Federal sources indicate that Congress decided to do so after receiving a number of comments that argued for the establishment of "special deposit procedures to mitigate the alleged uncertainties associated with depositing material containing trade secrets in a public office."[98] One additional constituency that was particularly focused on gaining dual protection involved standardized test preparers, who desired the ability to reuse their questions over multiple rounds of testing, but still keep the questions secret.[99] As the Register of Copyrights, Ralph Oman explained around that time:

> The Office originally asked for [protectability of] source code, because that best represents the copyrightable authorship. But many copyright owners say that the source code version of a program contains valuable trade secrets. . . . So the Office gave special relief to allow registration without disclosing trade secrets. Usually, we accepted an abbreviated deposit or a deposit with the trade secret material blocked out.[100]

There were other strategic reasons that weighed in favor of a dual system. As Zimmerman explains,

> By securing the source code behind a wall of secrecy, owners could get remedies for breach where access to the product was granted only sparingly and conditionally. But designers of software for PCs could not be sure that courts would treat their programming devices and choices as "secrets" once thousands, even millions, of copies of the programs embodying them were being sold (albeit in the impenetrable form of object code). Being able to claim copyright was a kind of legal insurance policy against the risk that a court might refuse to recognize the existence of trade secrets in software distributed to the public at large.[101]

ANTITRUST BULL. 651, 654 (1998) (recognizing the role of courts in maintaining the proper boundaries of copyright law).

98 Registration of Claims to Copyright Deposit Requirement for Computer Programs Containing Trade Secrets and for Computer Screen Displays, 54 Fed. Reg. 13,173, 13,173 (Mar. 31, 1989) (to be codified at 37 C.F.R. pt. 202).

99 Zimmerman, *supra* note 83, at 311.

100 Ralph Oman, *Software as Seen by the U.S. Copyright Office*, 28 IDEA 29, 30 (1987).

101 Zimmerman, *supra* note 83, at 311.

The Copyright Office, rather than Congress, decided to step in to solve the problem. Instead of requiring total deposit of the source code, the Copyright Office decided to require registrants to file the first and last twenty-five pages (or equivalent) of source code with the trade secret sections blocked out, so long as they were "proportionately less than the material remaining, and the deposit reveals an appreciable amount of original computer code."[102] In one of the few court challenges to address these special deposit requirements, the Seventh Circuit held that these specialized rules did not require public disclosure, and the Copyright Office was well within its purview of discretion in designing specialized rules for secret, copyrighted material.[103]

Yet this shift toward accommodation, I would argue, represented a contradiction in terms. The deposit requirements were historically motivated to promote access to the public; whereas the administrative tolerance for closed code was essentially designed to enable circumvention of disclosure altogether.[104] As a result, source code remains, even to this day, marred by its underlying incoherence between its expression as a (potentially public) authorial creation and its function as a closely held trade secret.

2. *Copyrighting Code*

While most software shops behaved collaboratively in the early years, relying on mostly contract and trade secrecy, that began to change in the early 1980s, when AT&T began to

[102] Registration of Claims to Copyright Deposit Requirement for Computer Programs Containing Trade Secrets and for Computer Screen Displays, *supra* note 98, at 13,176; *see also* U.S. COPYRIGHT OFFICE, COMPENDIUM OF U.S. COPYRIGHT OFFICE PRACTICES § 1509.1(C)(4)(d) (3d ed. 2017), https://www.copyright.gov/comp3/docs/compendium.pdf [https://perma.cc/73VY-44XM] (detailing the U.S. Copyright Office's instructions on the appropriate method for blocking out source code that contains trade secret material); Joseph Potvin, *How Is Copyright Relevant to Source Data and Source Code?*, TECH. INNOVATION MGMT. REV. (Feb. 2008), https://timreview.ca/article/121 [https://perma.cc/NC3Q-7WS7] (outlining how copyright law relates to source data and source code); Scott Bell, Aly Dossa & Timothy M. Smith, *To Protect Your Source Code, Treat It Like Intellectual Property*, SOFTWARE DEV. TIMES (July 12, 2011), https://sdtimes.com/intellectual-property/to-protect-your-source-code-treat-it-like-intellectual-property/ [https://perma.cc/XEW5-5HEM] (differentiating the protections for source code between trade secrets, copyrights, and patents).

[103] *See* Zimmerman, *supra* note 83, at 312 (discussing *Nat'l Conference of Bar Examiners & Educ. Testing Serv. v. Multistate Legal Studies, Inc.*, 692 F.2d 478 (7th Cir. 1982)).

[104] *See id.* at 313 (discussing how deposit requirements became loosened after fixation, rather than publication, became the focus of protection, and also due to space considerations at the Library of Congress).

threaten litigation to enforce its rights to Unix, an operating system that could run on multiple platforms. In response, Richard Stallman of the MIT Artificial Intelligence Laboratory started the Free Software Foundation, which aimed to distribute code openly and with restrictions in place to preclude assertions of proprietary control.[105]

As the Unix dispute demonstrated, the question of how source code and its secrecy intersected with intellectual property began to take on more importance before courts, the Copyright Office, and Congress. Today, despite the initial application of the rule of doubt in the case of software protection under copyright, it is well settled that copyright law protects the original, literal elements of both a program's source code and its object code.[106] Object code, too, is protectable

---

[105] There is a vast literature exploring the dynamics of the open source movement. *See, e.g.*, Greg Madey et al., *The Open Source Software Development Phenomenon: An Analysis Based on Social Network Theory*, AMCIS 2002 PROCS. 247 (2002) (discussing the way the open source software community works in order to improve reliance on open source software); Joachim Henkel, Simone Schöberl & Oliver Alexy, *The Emergence of Openness: How and Why Firms Adopt Selective Revealing in Open Innovation*, 43 RES. POL'Y 879, 879–90 (2014) (discussing cooperation in the open source software community); ERIC STEVEN RAYMOND, THE CATHEDRAL AND THE BAZAAR (2000), http://www.catb.org/esr/writings/cathedral-bazaar/cathedral-bazaar [https://perma.cc/LXF6-WNKG] (discussing open source development); Brian Fitzgerald, *The Transformation of Open Source Software*, 30 MIS Q. 587, 587 (2006) (arguing that the open source software movement has shifted from a "proprietary-driven model" to "a more mainstream and commercially viable form"); ROD DIXON, OPEN SOURCE SOFTWARE LAW (2004) (introducing the legal framework that has evolved to support the open source software community); Michael Schwarz & Yuri Takhteyev, *Half a Century of Public Software Institutions: Open Source as a Solution to Hold-Up Problem*, 12 J. PUB. ECON. THEORY 609 (2010) (arguing that proprietary software causes underinvestment in complementary products due to fears of hold up, and using this thesis to explain the success of open source in software development platforms like operating systems); Jeevan Jaisingh, Eric W.K. See-To & Kar Yan Tam, *The Impact of Open Source Software on the Strategic Choices of Firms Developing Proprietary Software*, 25 J. MGMT. INFO. SYS. 241 (2014) (comparing the effect of open source software on the marketplace for software innovation); Eric von Hippel, Open Source Software Projects as User Innovation Networks (unpublished manuscript) (draft on file with author) (studying conditions for user innovation); Maxim V. Tsotsorin, Comment, *Open Source Software Compliance: The Devil Is Not so Black as He Is Painted*, 29 SANTA CLARA COMPUTER & HIGH TECH. L.J. 559 (2013) (exploring dimensions of open source software compliance in licensing); V.K. Unni, *Fifty Years of Open Source Movement: An Analysis Through the Prism of Copyright Law*, 40 S. ILL. U. L.J. 271 (2016) (providing a broad overview of the history of the open source software movement); Jonathan Zittrain, *Normative Principles for Evaluating Free and Proprietary Software*, 71 U. CHI. L. REV. 265 (2004) (offering a framework for assessing the value of free and proprietary software).

[106] *See* Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 702 (2d Cir. 1992) ("It is now well settled that the literal elements of computer programs, i.e., their source and object codes, are the subject of copyright protection."). The Compendium of U.S. Copyright Office Practices further explains that it "considers

under copyright.[107] The argument, as it goes, follows this reasoning: "Since source code is copyrightable, and since source code can readily be translated into object code, object code must also be copyrightable."[108]

As Samuelson explains, the early cases that followed the 1980 Amendments focused on either the copying of audiovisual elements, code, or both.[109] For cases of line-by-line copying of source and object code, i.e., literal infringement, copyright served as a useful vehicle of protection.[110] The basic case establishing copyright infringement for the literal elements of program code was *Apple Computer, Inc. v. Franklin Computer Corp.*, in which Franklin copied, verbatim, Apple's operating system and several application programs.[111] While Franklin did not dispute the question of appropriation, it argued that Apple's operating system was not protectable under copyright, because unlike books or literary works, code was not intended to be read by a human.[112]

While the earliest code-related cases were relatively straightforward cases of misappropriation,[113] over time, the cases that raised more complexity involved "nonliteral" in-

source code to be the best representation of the copyrightable authorship in a computer program" for the purposes of examination, particularly because object code cannot be examined since it is unintelligible to humans. U.S. COPYRIGHT OFFICE, *supra* note 102, § 1509.1(C).

[107] While there was some initial trepidation over its copyrightability, due to its functionality, its protection is now well settled. LEMLEY ET AL., *supra* note 34, at 35.

[108] *Id.* at 37 (citing Dan L. Burk, *Software as Speech*, 8 SETON HALL CONST. L.J. 683, 687–88 (1998) (noting that courts have rejected arguments that source code is not copyrightable)).

[109] Samuelson, *supra* note 14, at 12.

[110] Arsenault, *supra* note 73, at 138–39.

[111] Apple Comput., Inc. v. Franklin Comput. Corp., 714 F.2d 1240 (3d Cir. 1983).

[112] LEMLEY ET AL., *supra* note 34, at 33–34. Indeed, the court's argument mirrored, almost perfectly, the observations offered by one CONTU Commissioner, who wrote in his dissent that "[p]rograms are profoundly different from the various forms of 'works of authorship' . . . [which] have always been intended to be circulated to human beings and to be used by them—to be read, heard, or seen, for either pleasurable or practical ends." *Id.* at 36–37 (quoting FINAL REPORT OF THE NATIONAL COMMISSION ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS 28 (1978) [hereinafter CONTU FINAL REPORT]). Yet despite these arguments, the court firmly concluded that object code could be protectable, reasoning that section 101 included an expansive list of categories of literary works, including those that comprised "numbers, or other . . . numerical symbols or indicia." *Apple Comput., Inc.*, 714 F.2d at 1247 (quoting 17 U.S.C. § 101 (1982)).

[113] *See, e.g.*, Cadence Design Sys., Inc., v. Avant! Corp., 125 F.3d 824 (9th Cir. 1997) (software manufacturer brought an action against a competitor for misappropriation of trade secrets); Engenium Sols., Inc. v. Symphonic Techs., Inc., 924 F. Supp. 2d 757 (S.D. Tex. 2013) (same).

fringement claims—the program's structure, its sequence, its organization, including some of the various steps that a programmer might take prior to even drafting the code itself.[114] The need for a theory to address those cases became even more prevalent as more and more cases of appropriation made their way to the courts.[115]

For example, structural, nonliteral claims proliferated throughout the courts, beginning with landmark cases like *Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc.*, culminating in cases involving the "structure, sequence, and organization" (SSO).[116] *Whelan* marked a watershed shift in the area of software protection, because it represented the first of a few cases that readily extended copyrightability to structure and organization, and to other nonliteral elements.[117] Later, courts began to narrow the breadth of *Whelan*'s applicability, articulating tests like *Altai* that encouraged courts to separate out unprotectable elements by first identifying which parts of the software comprise abstract ideas (as divorced from expression), then to filter out all unprotectable elements (like elements from the public domain), and then finally to compare all remaining elements to determine infringement.[118]

Afterward, courts and scholars tended to focus mostly on nonliteral forms of infringement, like the program's structure and organization, including flow charts, intermodular relationships, parameter lists, and macros.[119] Literal forms of infringement, such as source or object code appropriation, remained a deceptively simplified area of intellectual property

114 *See* Richard Raysman & Peter Brown, *Copyright Infringement of Computer Software and the 'Altai' Test*, 235 N.Y. L.J., May 9, 2006, at 1–2 (discussing cases).

115 For such cases, the operable question became how far courts were prepared to depart from the literal expression of the code to protect other elements under copyright principles. *See* Arsenault, *supra* note 73, at 140 (discussing this in more detail).

116 Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc., 797 F.2d 1222, 1240 (3d Cir. 1986).

117 *See id.* at 1239 (explaining that the structure of a program is part of the expression, not the idea, of that program); *see also* SAS Inst., Inc. v. S&H Comput. Sys., Inc., 605 F. Supp. 816, 830 (M.D. Tenn. 1985) (recognizing the extent that a competitor copied the organizational and structural details of SAS); Apple Comput., Inc. v. Microsoft Corp., 35 F.3d 1435, 1445 (9th Cir. 1994) (recognizing user interfaces, input formats, and output reports); Eng'g Dynamics, Inc. v. Structural Software, Inc., 26 F.3d 1335, 1342–43 (5th Cir. 1994) (endorsing the abstraction-filtration-comparison method).

118 Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 693 (2d Cir. 1992).

119 Raysman & Brown, *supra* note 114, at 1 (quoting *Comput. Assocs. Int'l, Inc.*, 982 F.2d at 702).

protection, even though cases continued to quietly percolate through the courts.

D. The Continuing Overlap Between Copyright and Trade Secrecy

Nevertheless, despite the potential role of copyright protection, secrecy continued to dominate, even though one of the most significant developments in the history of software was the rise of the open source movement, which emerged out of a distrust of software secrecy in the 1990s.[120] The movement generally comprises a combination of two core principles: the first involves the visibility of source code; the second involves the right to create relatively unencumbered derivative software for any purpose, including education or commercial.[121] Since the 1990s, the open source movement has also given rise to a growth of collaborative activity, where commercial and open source endeavors bundle cooperatively-developed software with proprietary code.[122]

Today, in the context of proprietary software, most companies market products in object code format only; the source code remains firmly in the developer's hands, secluded from the public and only shared upon the execution of a contract to protect its secrecy.[123] There are many reasons for this, not the least of which is secrecy.[124] Object code is easier to install, since file sizes are smaller, and preserves the secrecy of the source code.[125]

But this makes source code an awkward fit for copyright law as a result. As Gibson has explained, too much private control over copying and dissemination could deny the public

120 Wirth, *supra* note 38, at 37. For a great discussion of issues facing the open source community, see Philip J. Weiser, *Law and Information Platforms*, 1 J. ON TELECOMMS. & HIGH TECH. L. 1, 6–16 (2002); see also *supra* note 105.

121 *See* Diomidis Spinellis & Clemens Szyperski, *How Is Open Source Affecting Software Development?*, IEEE SOFTWARE, Jan.-Feb. 2004, at 28, 29 (recognizing these two core principles); *Source Code Definition, supra* note 45. As one commentator observes, "Open source appeared as the welcome alternative to industrial hegemony and abrasive profit, and also against helpless dependence on commercial software." Wirth, *supra* note 38, at 37.

122 Lerner & Tirole, *supra* note 75, at 7.

123 Gibson, *supra* note 20, at 175.

124 *See Source Code Definition, supra* note 45.

125 *Id.*; *see also* Mark A. Lemley & David W. O'Brien, *Encouraging Software Reuse*, 49 STAN. L. REV. 255, 272–73 (1997) (discussing the reliance on trade secrecy in software); Data Gen. Corp. v. Grumman Sys. Support Corp., 825 F. Supp. 340, 359 (D. Mass. 1993), *aff'd*, 36 F.3d 1147 (1st Cir. 1994) (noting that object code distribution does not disclose trade secrecy).

access to goods and raw materials necessary for innovation.[126] But too little private control risks underproduction.[127] For this reason, the traditional architecture of copyright offers a limited scope of protection to the owner, but tempers this private right with a number of dedicated entitlements to the public, including a finite term of protection (which then dedicates the work to the public domain), the first sale doctrine, or fair use protection in certain cases.[128]

In other words, as Gibson notes, traditional copyright law is intended to force a choice onto authors: if they keep the work from the public, they forego profiting from it; or, they can bring their creative works to the public and enable the public to see the expression.[129] In most cases of copyrighted works like literary works, motion pictures, or musical recordings, the author has a sustained interest in publication, since it promotes sales of the underlying work. The value of the copyrighted work is thus inherently tied to the expectation of publication.

The exact opposite is true in the case of source code secrecy: its very value lies in its seclusion from the public. The expected unification between publication and marketability simply does not exist in the context of software, where secrecy represents no obstacle to marketability.[130] As Gibson further observes:

> With software, however, we have a copyrighted work whose unique architecture allows its author to profit without revealing either its creative expression or its ideas to the purchaser. The software developer thus receives the benefit of copyright protection—the right to sue anyone who engages in unauthorized reproduction or adaptation of the program—without conferring the corresponding benefit on the rest of us. Whatever ideas exist in the creative source code of a computer program remain with the developer; all the public encounters is an impenetrable and unrevealing string of ones and zeroes.[131]

In other words, as Gibson concludes, "[o]nly with software may authors have their cake and eat it too."[132]

In contrast to object code, which has a public nature, source code's content can be kept secret, even without any

---

126 Gibson, *supra* note 20, at 170–71.
127 *Id.*
128 *Id.*
129 *Id.* at 178.
130 *Id.*
131 *Id.* at 181 (footnote omitted).
132 *Id.* at 178.

detrimental effect to its marketability.[133] Further, the Supreme Court has suggested that there is no disclosure obligation under copyright.[134] As Laura Heymann notes,

> Copyright law may be justified by the ultimate goal of disseminating works of authorship to the public, but since the move in the 1976 Copyright Act from publication to fixation as the triggering event for protection, the diary tucked away in a desk drawer receives just as much protection as the best-selling novel.[135]

In the context of software, this means that both copyright and trade secret law can overlap—producing something that can be protected because of its content, but also kept from the public because it is a trade secret.[136]

Moreover, there are other ironies to this situation: copyright law protects object code, which manifests no creativity but is largely functional in nature, whereas trade secret law (traditionally the legal vehicle for protecting functional processes) has now become the vehicle to protect source code (despite its creativity).[137] The issue of functionality in copyrightable processes has troubled courts for years, starting back in 1880 with *Baker v. Selden*, when the Supreme Court rejected the copyrightability of an accounting system on the grounds that it would confer protection over the system or process itself.[138] The same idea of limiting copyrightability for actual processes, such as those found in software architecture, as operations within modules, or as algorithms, has remained a consistent source of judicial attention.[139] Similarly, courts have expressed concerns over the copyrightability of facts, and have defined "facts" in the software context to include not only

[133] *Id.* at 173.

[134] *See* Laura A. Heymann, *Overlapping Intellectual Property Doctrines: Election of Rights Versus Selection of Remedies*, 17 STAN. TECH. L. REV. 239, 257–58 (2013) (noting that in *Eldred v. Ashcroft*, the Court noted that "our references to a *quid pro quo* typically appear in the patent context") (quoting Eldred v. Ashcroft, 537 U.S. 186, 216 (2003)).

[135] *Id.*

[136] *Id.*

[137] *See id.*

[138] Baker v. Selden, 101 U.S. 99, 104 (1880), *superseded by statute*, Copyright Act of 1976, Pub. L. No. 94-553, 90 Stat. 2541 (1976). *See also generally* Pamela Samuelson, *The Story of* Baker v. Selden*: Sharpening the Distinction Between Authorship and Invention*, *in* INTELLECTUAL PROPERTY STORIES 159 (Jane C. Ginsburg & Rochelle Cooper Dreyfuss eds., 2006) (detailing the impact of the case on copyright law analyses).

[139] *See* Gates Rubber Co. v. Bando Chem. Indus., Ltd., 9 F.3d 823, 837 (10th Cir. 1993). *See generally* Pamela Samuelson, Benson *Revisited: The Case Against Patent Protection for Algorithms and Other Computer Program-Related Inventions*, 39 Emory L.J. 1025 (1990).

parts of data structures, but also material that is literally expressed within source or object code as well.[140] Here, when material that is in the public domain comprises part of a program, courts have advocated the need to filter out the unoriginal parts of a program by relying on both the merger and *scènes à faire* doctrines to aid in the filtration.[141]

Yet—paradoxically—the intersection of trade secrecy with copyright in software runs contrary to the value of disclosure, further impoverishing the public domain that is at the heart of copyright's architecture.[142] The result is that our existing regime fails to ensure the availability of public domain aspects of software, and precludes evaluation of the protectability of the code altogether.[143] The resulting irony, Gibson notes, is particularly striking: "[T]he law tells us that software comprises more public domain elements than other copyrighted works, but the architecture of closed code protects software more thoroughly than any of its copyrighted counterparts."[144]

II
THE SHIFTING BOUNDARIES OF SOFTWARE PATENTABILITY

As the above section suggested, software has a hybrid character: like other copyrighted works, it expresses various concepts, but, like a patented invention, it has the power to physically implement those ideas only with the assistance of a computer.[145] This potentially rivalrous relationship between copyright and patent to software produced an especially lively debate from the mid-1980s to 1990s.[146] The issue rightfully led commentators, then-Professor Stephen Breyer among them, to question the need to extend copyright to computer programs on the grounds that there were already substantial incentives in place to encourage their production.[147] Although some of those controversies waned a bit after the 1990s when a few appellate courts began to narrow the scope of copyrightability in software due to its functionality, the debate has more

140 *Gates Rubber Co.*, 9 F.3d at 837–38.

141 *Id.*

142 Gibson, *supra* note 20, at 178.

143 *Id.*

144 *Id.* at 181.

145 Gregory J. Maier, *Software Protection—Integrating Patent, Copyright and Trade Secret Law*, 69 J. PAT. & TRADEMARK OFF. SOC'Y 151, 151 (1987).

146 Samuelson, *supra* note 14, at 1.

147 *See* Breyer, *supra* note 29, at 344; Samuelson, *supra* note 14, at 5. For a modern-day response to Justice Breyer's article, see Pamela Samuelson, *The Uneasy Case for Software Copyrights Revisited*, 79 GEO. WASH. L. REV. 1746 (2011).

recently picked up steam with the advent of the *Oracle America, Inc. v. Google Inc.* case, which reinvigorated the overlap between copyright and patent claims for program interfaces.[148] The patentability of code also suffers from similar issues that, like the copyright regime discussed above, tends to push developers toward the domain of trade secrecy.[149]

If copyright's regime is directed toward maximizing the benefits of publication, the patent system is equally motivated toward maximizing the benefits of disclosure.[150] Both of these policy goals contradict the comparative value of secrecy, however, in the context of source code, and therein lies the problem.

Of course, law is also not the only reason to opt for secrecy. The market also tends to support similar choices.[151] Surveys have shown that company executives rank trade secrets as the area of primary importance in their intellectual property portfolios.[152] Secrecy also becomes incredibly attractive when the nature of the invention is more easily able to be kept secret, as in software, unlike industries like pharmaceuticals or con-

---

[148] Samuelson, *supra* note 14, at 2 (referring to Oracle Am., Inc. v. Google Inc., 750 F.3d 1339 (Fed. Cir. 2014), *cert. denied*, 135 S. Ct. 2887 (2015)).

[149] For more discussion of the trade secrecy/patent interface, see generally Andrew Beckerman-Rodau, *The Choice Between Patent Protection and Trade Secret Protection: A Legal and Business Decision*, 84 J. PAT. & TRADEMARK OFF. SOC'Y 371 (2002) (discussing the legal and business considerations for an inventor in choosing between reliance on patent or trade secret law); David S. Almeling, *Seven Reasons Why Trade Secrets Are Increasingly Important*, 27 BERKELEY TECH. L.J. 1091 (2012) (detailing the rise in trade secret reliance); Michael Risch, *Trade Secret Law and Information Development Alternatives*, *in* THE LAW AND THEORY OF TRADE SECRECY: A HANDBOOK OF CONTEMPORARY RESEARCH, *supra* note 83, at 167–76 (exploring different incentives underlying trade secret law versus patent law); Bruce T. Atkins, *Trading Secrets in the Information Age: Can Trade Secret Law Survive the Internet?*, 1996 U. ILL. L. REV. 1151, 1174–75 (noting the decline of trade secret protection in the internet era).

[150] Stuart J.H. Graham, Robert P. Merges, Pamela Samuelson & Ted Sichelman, *High Technology Entrepreneurs and the Patent System: Results of the 2008 Berkeley Patent Survey*, 24 BERKELEY TECH. L.J. 1255, 1317 (2009); *see also* Katherine J. Strandburg, *What Does the Public Get? Experimental Use and the Patent Bargain*, 2004 WIS. L. REV. 81, 111 (discussing the benefits of disclosure under the patent system).

[151] *See* Andrew A. Schwartz, *The Corporate Preference for Trade Secret*, 74 OHIO ST. L.J. 623, 624 (2013).

[152] *See* Hamid Sakaki & Karn Thapar, *Trade Secrets Protection and Corporate Tax Avoidance*, J. ACCT. & FIN. (forthcoming) (manuscript at 3) (discussing a National Science Foundation survey that found that certain for-profit companies across all industries ranked trade secrets as the most important kind of intellectual property for their businesses). In 2008, a Berkeley Patent Survey revealed that in industries like software, internet, manufacturing, and chemical processing, patenting was perceived to be far less important as a means to ensure a competitive advantage. *See* Graham et al., *supra* note 150, at 1260; *see also* J. Jonas Anderson, *Secret Inventions*, 26 BERKELEY TECH. L.J. 917, 927 (2011).

sumer products, where the nature of an invention is more accessible to the public.[153]

Even aside from these market-based reasons to opt for secrecy, the law has taken a curious path that has only further served to quietly marshal resources toward trade secrecy. The Court, for example, has entertained its own set of debates over whether software is patentable, first rejecting the prospect, then reversing itself, only to return to a more cynical view over patentability more recently.[154] As I argue below, the result of these shifts has only underscored the comparative attractiveness of trade secrecy, largely at a cost to the public interest in transparency.

A. Patentability vs. Secrecy

In general, patent and trade secret protection are ideally supposed to be mutually exclusive, since the patent system does everything it can to discourage secrecy.[155] Rather, the patent system uses the powerful grant of a monopoly power as the proverbial carrot in order to compel inventors to reveal the nature of their inventions to the public.[156] This way, the patent grant forces society to essentially "pay" for secrets which would be otherwise unavailable to them by making the nature of the invention informationally available upon conferral of the patent and by enabling the public to practice the invention after the term of protection has expired.[157]

The idea that an applicant is supposed to "elect" between patent and trade secrecy is a powerful, meaningful aspect of our system of intellectual property. And disclosure to the public is a core goal of patent law, as Jeanne Fromer has argued in her work, because it promotes follow-on innovation.[158] But in actuality, even in the context of software patents, the disclo-

[153] Anderson, *supra* note 152, at 927; *see also* Wesley M. Cohen et al., *Protecting Their Intellectual Assets: Appropriability Conditions and Why U.S. Manufacturing Firms Patent (or Not)* 3, 24 (Nat'l Bureau of Econ. Research, Working Paper No. 7552, 2000), http://ssrn.com/abstract=214952 [https://perma.cc/M7NN-3CEQ]. For an excellent study of the role of secrecy in startups, see David S. Levine & Ted Sichelman, *Why Do Startups Use Trade Secrets?*, 94 NOTRE DAME L. REV. 751, 753 (2018).

[154] *See infra* subpart II.B.

[155] Anderson, *supra* note 152, at 928.

[156] *Id.*

[157] *Id.* at 929.

[158] Jeanne C. Fromer, *Patent Disclosure*, 94 IOWA L. REV. 539, 541 (2009). Disclosure, she argues, stimulates productivity in two ways: first, by enabling society to use the information after the patent expires; second, by enabling inventors to design around the invention or to conceive of new inventions even during the patent term. *Id.* at 548–50.

sure requirements are so relaxed that they are minimally effective.[159] There is no requirement for source code disclosure, and because a patent might only cover only one small portion or module of the code, trade secrecy can still attach to the rest of the product.[160] And software distribution models are set up, essentially, to protect secrecy at all costs; even when these practices migrate to the cloud, for example, they still take pains to protect secrecy above anything else.[161] As Greg Vetter explains:

> Given the opportunity to continue to rely on trade secrecy, most proprietary software vendors will continue to do so and obtain patents when it matters strategically. Patent protection is much more costly for the software product vendor than the other modes of protection. It requires a parallel stream of activity alongside the development of the patent. . . . In contrast, trade secret protection and copyright protection in the software is essentially without additional cost given that the business practices of the software industry give these modes of protection by default.[162]

But even outside of the particular question of software patentability, there are several fundamental differences between a patent and a trade secret that may—generally—compel seclusion over disclosure.[163] First, consider duration. A trade secret can be limitless in its duration, as long as it remains a secret, in contrast to the twenty-year protection afforded to patents.[164] The twenty-year protection period is of little value in the software industry because software typically becomes obsolete by the time a patent even issues.[165]

Second, the process of obtaining a patent can be onerous, time consuming, labor intensive, and expensive.[166] Patent attorneys must disclose the nature of the invention and prove that it meets the requirements of novelty, nonobviousness, and utility.[167] After an initial application is filed (which is often rejected), the inventor must engage in a lengthy back-and-forth

159 Greg R. Vetter, *Are Prior Use Rights Good for Software?*, 23 TEX. INTELL. PROP. L.J. 251, 305 (2015).
160 *Id.*
161 *Id.*
162 *Id.* at 306.
163 Anderson, *supra* note 152, at 923.
164 *Id.*
165 *See Patents, Copyrights, and Your Software Innovation*, U. WASH., https://comotion.uw.edu/what-we-do/patents-copyrights-and-your-software-innovation/ [https://perma.cc/VT3P-7JEU] (making this observation).
166 *See* Anderson, *supra* note 152, at 924.
167 *Id.*

process with a patent examiner, often necessitating amendments, further filings, and complicated negotiations with the patent office.[168] With patentability, the process is uncertain, may lead to years of time-consuming and costly amendments, and may not always result in a protectable patent.[169]

Finally, even after a patent is granted, they are difficult to enforce. Patent litigation can be extremely costly and expensive, and inventors are required to constantly monitor the market for possible infringement, which can often be complicated depending upon the nature of the invention and the ease of monitoring.[170]

By contrast, trade secrecy reduces significant administrative and judicial costs associated with acquiring a patent.[171] There is no central office to register trade secrets; a mere assertion of trade secrecy is all that is needed in order to keep that information from the public.[172] Given the expense of time and resources that are required to acquire a patent, plus the onerous costs of patent litigation, many inventors rationally choose the trade secret route.[173] "By choosing secrecy," one author maintains, "inventors avoid the cost of obtaining a patent, and the risky, costly business of patent enforcement."[174] In the case of source code protection, the uncertainty of patent protection, especially in a post-*Alice* world, can push inventors toward the rational belief that the code is much more valuable as a secret than as a patented invention—thus eliminating the comparable costs of seeking a patent.[175] If the costs of patenting an invention are higher, and the grant of protection uncertain, inventors may rationally opt for a trade secret solution instead.

Amplifying this point, Mark Lemley has argued that source code presents particular characteristics that make trade secrecy even more desirable.[176] Because source code cannot be discerned by purchasing the product (unless it is reverse engineered), trade secrecy gives owners an advantage because it

---

168 *Id.* at 925.

169 *See* Himanshu S. Amin, *The Lack of Protection Afforded Software Under the Current Intellectual Property Laws*, 43 CLEV. ST. L. REV. 19, 22–23 (1995).

170 Anderson, *supra* note 152, at 925.

171 *Id.* at 920.

172 *Id.*

173 *Id.*

174 *Id.* at 925.

175 *Id.* at 920 (discussing these benefits in trade secrecy generally).

176 *See* Mark A. Lemley, *The Surprising Virtues of Treating Trade Secrets as IP Rights*, 61 STAN. L. REV. 311, 339–40 (2008); *see also* Risch, *supra* note 149, at 165–81 (making a similar point).

allows them to keep the invention secret, whereas patents can be invalidated or designed around.[177] For this reason, when secrecy is possible, it is often chosen over patent protection.[178]

At the same time, however, trade secrecy is a weaker form of protection than patent protection, because a trade secret can be destroyed by independent invention or reverse engineering, both of which do not serve as defenses in our patent system.[179] In situations where independent discovery or reverse engineering is possible or likely, patent protection may be the better choice.[180] But in cases where independent discovery or reverse engineering is less likely, trade secrecy may be a more preferable route.[181]

Many of these same characteristics have led others, Robert Bone most prominently, to question the value of trade secrecy, arguing that justifications that focus on the shortcomings of the patent system operate like "a stop-gap measure, like a rag used to plug a hole in a pipe that actually requires a more extensive repair job."[182] While trade secrecy might incentivize owners to share information with potential business partners, it does very little to encourage sharing with the public or to encourage follow on innovation.[183] While Lemley argues that trade secrecy reduces the need to overinvest in secrecy, because it acts as a substitute for investments in physical secrecy, I would point out that software is an example to the contrary, because physical seclusion is of very low cost.[184]

But trade secrecy is not a costless enterprise, either. Inventors who take the secrecy route are also required to engage in self-help measures to deter discovery, physically protect the trade secret, and administer a maze of nondisclosure and employee confidentiality agreements.[185] These agreements, as

177 *See* Lemley, *supra* note 176, at 340.

178 *See id.*

179 *See id.*

180 *See id.* at 340–41; Gregory V. Novak & Matthew Frontz, *Tipping the Scales: Weighing IP Protection Options Post-DTSA and Post-*Alice, TEX. LAW., Dec. 2016, at 42, 42.

181 Lemley, *supra* note 176, at 340–41.

182 Robert G. Bone, *The (Still) Shaky Foundations of Trade Secret Law*, 92 TEX. L. REV. 1803, 1814 (2014) [hereinafter Bone, *Shaky Foundations*]; *see also* Robert G. Bone, *A New Look at Trade Secret Law: Doctrine in Search of Justification*, 86 CALIF. L. REV. 241, 265–70 (1998) [hereinafter Bone, *A New Look*] (considering whether trade secret law improves efficiency).

183 Bone, *Shaky Foundations*, *supra* note 182.

184 *See* Lemley, *supra* note 176, at 333–34.

185 Anderson, *supra* note 152, at 925.

well, are not always enforceable by courts, introducing added risk factors to the cost-benefit continuum.[186]

Despite the costs of secrecy, there are many reasons for its preferability from a software developer perspective, not always involving profit and protection.[187] Because trade secret protection can extend to both the underlying design concepts of a computer program and its expression of those concepts, it is considered to be particularly suitable for software.[188] Physically, source code secrecy can easily be maintained, in contrast to other inventions (like an improved pop-top soda can), which are disclosed by their public nature.[189] Inventions that can be easily shielded from public view or are difficult to reverse engineer, like source code, can be a particularly attractive fit for trade secrecy.[190]

Other motivations for secrecy can also stem from wanting to protect against security-related risks like malware and other forms of viruses.[191] Still other reasons are motivated toward authorial self-protection; that is, to avoid the risk that disclosure of the source code could expose developers to charges of plagiarism if the code is not considered to be sufficiently original, or even to place obstacles regarding being used as evidence in legal decisions.[192] Or, the desire to protect the code might also stem from concerns that clients may try to modify the source code for their own purposes, instead of contacting the developer directly.[193] And there may also be concerns about revealing the internal commentary inserted by programmers within the source code, which can often be colorful or offensive in nature.[194] All of these reasons further undercore the attractiveness of trade secrecy, particularly in light of the shifting sands of patentability, which I discuss below.

B. The Rise and Fall of Software Patentability

In the mid-1960s, the U.S. Patent Office first opposed issuing patents not only for programs but also for processes that

---

186 *Id.*

187 *See Source Code Definition*, *supra* note 45.

188 Arsenault, *supra* note 73, at 136.

189 Anderson, *supra* note 152, at 925.

190 *See* Jeanne C. Fromer, *A Legal Tangle of Secrets and Disclosures in Trade:* Tabor v. Hoffman *and Beyond*, *in* INTELLECTUAL PROPERTY AT THE EDGE: THE CONTESTED CONTOURS OF IP 271 (Rochelle Cooper Dreyfuss & Jane C. Ginsburg eds., 2014).

191 *See Source Code Definition*, *supra* note 45.

192 *Id.*

193 *Id.*

194 *Id.*

were embodied in programs, on the ground that the former category were authorial works and the latter were mental processes.[195] Further, there was no established body of prior art in place for the Patent Office to conduct comparisons to previous advances in the field to determine things like novelty and nonobviousness.[196] It further reasoned that it would be extremely difficult to compile a suitable database of prior art and design a system of classification on what it had not yet investigated.[197] No centralized patent registry existed for software, nor does one exist today.[198] As a result, when software receives patent protection, it may be very difficult to protect due to the difficulties in detecting infringement.[199]

All of these rationales collectively made patenting less desirable than, say, copyright for practical reasons. But courts did not always share the Patent Office's early reluctance, and began, albeit slowly, to open the door toward patentability. Consider this illustration: Back in 1972, in *Gottschalk v. Benson*, an applicant was unable to obtain a patent on a method to convert binary-coded-decimal (BCD) numbers into pure binary numerals, due to the fact that the algorithm was an abstract idea, rather than a process.[200] Yet the Court was careful to note, even at that time, that its holding should not be taken to suggest a complete preclusion of patents for computer programs.[201] It further characterized the debates over patentability as a "policy matter," suggesting the need for further legislative intervention to decide the issue.[202]

---

[195] *See* Samuelson, *supra* note 14, at 8.

[196] *See* Gibson, *supra* note 20, at 189.

[197] *See* Samuelson, *supra* note 14, at 8.

[198] *See* Gibson, *supra* note 20, at 189.

[199] *See Patents, Copyrights, and Your Software Innovation*, *supra* note 165 (making this observation).

[200] *See* 409 U.S. 63, 68 (1972) (describing applicant's claim as "so abstract and sweeping as to cover both known and unknown uses of the BCD to pure binary conversion"); *see also* Samuelson, *supra* note 139 (discussing *Gottschalk*).

[201] Because the mathematical formulas only worked with a computer, the court feared that a patent would "pre-empt the mathematical formula and in practical effect would be a patent on the algorithm itself." *Gottschalk*, 409 U.S. at 71–72.

[202] *See id.* at 72. Specifically, the Court also noted the position of the President's Commission on the Patent System, which rejected the idea that computer programs were patentable because of the lack of fit regarding subject matter and the inability to classify or search prior art. *Id.* "Without this search," the Commission concluded, "the patenting of programs would be tantamount to mere registration and the presumption of validity would be all but nonexistent." *Id.* The Court continued to refrain from extending patentability to software-related inventions in *Parker v. Flook*, a later case that raised similar issues of patentability. *See* Samuelson, *supra* note 14, at 9–15 (discussing Parker v. Flook, 437 U.S. 584 (1978)).

1. *The Opening of the Window of Patentability*

Eventually, a small window of patentability began to open in the 1981 case of *Diamond v. Diehr*,[203] which extended patentability to a process of curing rubber that relied, in part, on a computer program.[204] In that case, however, the Court extended protection to the program on the grounds that it represented only one element of the process, not because it could be protected on its own as a software patent.[205] Nevertheless, in the mid- to late 1980s, as Samuelson explains in her detailed account, it became clearer that the window for patentability began to open wider, and the Patent and Trademark Office began to issue more and more software patents.[206]

Although developers largely welcomed the rise of copyright and patentability involving software, the overlap between them raised critical questions regarding the accommodation of nonliteral forms of infringement.[207] Limiting copyrightability to source code would have been too narrow, but broadening copyrightability beyond source code risked intruding onto patent law's domain. "If copyright protection was only available to literal code, it would be easy to rewrite the same program design in noninfringing source code," Samuelson explained.[208] On the other hand, however, if copyright is considered more broadly, that is, if it "extended to the logic, design, structure, performance, or even the output of the computer program," this would risk giving the owner more patent-like protection, with a longer duration than patent protection and without meeting the comparably more rigorous requirements of patentability.[209] The ongoing instability over how to protect software, over time, led to a vigorous series of debates about whether the frameworks for nonliteral infringement were too broad and

203 450 U.S. 175, 184 (1981).

204 *See* Samuelson, *supra* note 14, at 9–15, 15 n.81 (discussing Diamond v. Diehr, 450 U.S. 175 (1981); and Maureen A. O'Rourke, *The Story of* Diamond v. Diehr*: Toward Patenting Software (Patents)*, *in* INTELLECTUAL PROPERTY STORIES, *supra* note 138, at 212–13).

205 *See* Samuelson, *supra* note 14, at 15 n.81. In fact, Samuelson argues that had Whelan not framed software copyrights so broadly, we might have seen even more of an upsurge. *Id.*

206 *See id.* at 15–16.

207 *See id.*

208 *Id.*

209 *See id.* at 16 (citing OFFICE of TECH. ASSESSMENT, INTELLECTUAL PROPERTY RIGHTS IN AN AGE OF ELECTRONICS AND INFORMATION 81 (1986)). Samuelson describes how, back in 1989, IP lawyers would "characterize nonliteral software structures as methods when they wanted to patent them and as SSO when asserting copyright." *Id.* at 40 n.258.

about the impact of software patents and copyrights on innovation.[210]

Eventually, courts began to narrow the scope of nonliteral infringement in cases like the Second Circuit's *Altai*[211] decision, which criticized *Whelan*[212] and was followed by other circuits.[213] *Altai*, in effect, produced a trend that led to greater discernment among courts in differentiating the roles of patent and copyright in protecting software.[214] But it also indirectly facilitated another outgrowth of software patenting, particularly due to the Federal Circuit's blessing of software patenting in the early 1990s.[215] In 1994, the Federal Circuit, in *In re Alappat*,[216] built on previous jurisprudence and, over the recommendations of the PTO, found that a "computer operating pursuant to software *may* represent patentable subject matter," in a case where the computer relied on an algorithm to transform a digital screen to display smooth waveforms in a digital oscilloscope.[217] There, the court explained that program instructions from software essentially transformed the machine from a "general purpose computer" into, in effect, a "special purpose computer" deserving of patentability.[218]

Looking back, if *Diehr* cracked a window to software patentability, *Alappat* opened it even further. And, after *Alappat*, the Federal Circuit decided *State Street Bank & Trust Co. v. Signature Financial Group, Inc.*,[219] which, in effect, did more than further open the window—it literally threw open the door to software patenting. *State Street* found that mathematical algorithms, previously dismissed as an abstract concept, could be patentable if they "transformed" a machine or were "performed" by a machine and provided "useful, concrete, and tangible" results.[220] This decision, more so than anything else,

210 *See id.* at 18.

211 Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 710 (2d Cir. 1992).

212 Whelan Assocs., Inc. v. Jaslow Dental Lab., Inc., 797 F.2d 1222, 1240 (3d Cir. 1986).

213 *See* Samuelson, *supra* note 14, at 19.

214 *See id.* at 21–22.

215 *See id. See generally* Alan D. Minsk, *The Patentability of Algorithms: A Review and Critical Analysis of the Current Doctrine*, 8 SANTA CLARA COMPUTER & HIGH TECH. L.J. 251, 277 (1992) (referencing the federal circuit cases).

216 33 F.3d 1526 (Fed. Cir. 1994).

217 *See* Fabio E. Marino & Teri H.P. Nguyen, *From* Alappat *to* Alice*: The Evolution of Software Patents*, 9 HASTINGS SCI. & TECH. L.J. 1, 4–5 (2017).

218 *In re Alappat*, 33 F.3d at 1566 n.28.

219 149 F.3d 1368 (Fed. Cir. 1998).

220 *See* Marino & Nguyen, *supra* note 217, at 4.

dramatically opened the door to software patenting, particularly in the world of business method patents.[221]

With *State Street*, and eventually *AT&T Corp. v. Excel Communications Marketing, Inc.*,[222] the golden era of software patents soon arrived. The PTO addressed this shift by classifying a specific type of patent for business methods, Internet, and software-related patents, known as a Class 705 patent.[223] By the 1990s and 2000s, companies were patenting software in droves compared to previous eras.[224] In 1998, there were 1,320 patent applications; by 2001, that number rose to nearly 8,000, peaking at over 10,000 applications in 2008.[225]

During this period, advocates of software patenting lauded the system's values of openness, interoperability, protection, and innovation due to its predication of disclosure and strong protection.[226] Yet, even then, software patents had their wide share of critiques.[227] In one representative example of this perspective, which came to be even more pronounced in later years, Simson Garfinkel, along with Richard Stallman and Mitchell Kapor, warned in 1991 that software patents were "being granted at an alarming rate," arguing that "most of the patents have about as much cleverness and originality as a recipe for boiled rice—simple in itself but a vital part of many sophisticated dishes."[228] To them, the patents covered everything from small and specific algorithms to techniques used in a wide variety of programs that were often used by others.

221 *See id.*

222 172 F.3d 1352 (Fed. Cir. 1999).

223 Class 705 includes a "generic class for apparatus and corresponding methods for performing data processing operations, in which there is a significant change in the data or for performing calculation operations wherein the apparatus or method is uniquely designed for or utilized in the practice, administration, or management of an enterprise, or in the processing of financial data." *See* Marino & Nguyen, *supra* note 217, at 6 (quoting the PTO's classification of Class 705).

224 *See* Samuelson, *supra* note 14, at 22.

225 *See* Marino & Nguyen, *supra* note 217, at 6–7 (citing PTO statistics and Starling Hunter's Article).

226 *See* Smith & Mann, *supra* note 31, at 256.

227 For critiques of software patenting, see Robert E. Thomas, *Debugging Software Patents: Increasing Innovation and Reducing Uncertainty in the Judicial Reform of Software Patent Law*, 25 SANTA CLARA COMPUTER & HIGH TECH. L.J. 191 (2008). *See also* James Bessen, *A Generation of Software Patents*, 18 B.U. J. SCI. & TECH. L. 241, 242 (2012) (challenging the benefits of software patenting in the software industry); Arti K. Rai, *Improving (Software) Patent Quality Through the Administrative Process*, 51 HOUS. L. REV. 503, 504 (2013) (arguing that software patents are of poor quality and outlining ways to improve them at the PTO level).

228 *See* Simson L. Garfinkel, Richard M. Stallman & Mitchell Kapor, *Why Patents Are Bad for Software*, ISSUES IN SCI. & TECH., Fall 1991, at 50, 51.

Because computer programs from 1991 were so complex, covering thousands of algorithms and techniques they posed enormous transaction costs for licensing, particularly when many of the newfound patents seemed overly broad.[229] The authors offered the example of a lawsuit against Apple for its violation of a patent that covered a specific technique for scrolling through a database. While apparently scrolling and display techniques were ubiquitous throughout software, separately, the patent at issue covered the combination of the two.

Aside from being overly broad at times, the length of time, plus the confidentiality of applications under review, made it very difficult for other parties to discern the likelihood of an application being granted.[230] It remained nearly impossible for applicants to search for prior art because the PTO had not yet developed a system for classifying algorithms and because the field of computer science literature is extraordinarily broad and hard to navigate. As a result, many patents were granted not because they were truly novel but because the examiner and the applicant may have been unaware of prior art on the subject.[231] The influx of so many patents meant that developers either rewrote code to avoid allegations of infringement or decided to avoid introducing new features entirely, thereby impeding innovation as a result. And their cost and complexity, not to mention the great amount of time they required, meant that many companies were shut out of the patenting process.

One report, examining patents over an eight year period ending in 1996, found that 46% of all patents were invalidated; when only software patents were considered, the number rose to two-thirds, attributable to the absence of a body of prior art, lower standards for non-obviousness, and PTO institutional pressures.[232] Other studies argued that software patents, far from encouraging innovation, actually led to more investment in building patent portfolios and enforcing them in court instead of research and development.[233] This was, the authors

[229] *See id.* at 52.

[230] *See id.*

[231] *See id.* at 53.

[232] *See* John R. Allison & Mark A. Lemley, *Empirical Evidence on the Validity of Litigated Patents*, 26 AIPLA Q.J. 185, 205–06, 217 (1998); Mark H. Webbink, *A New Paradigm for Intellectual Property Rights in Software*, 2005 DUKE L. & TECH. REV. 12.

[233] *See* JAMES BESSEN & ROBERT M. HUNT, THE SOFTWARE PATENT EXPERIMENT 2 (2004), www.researchoninnovation.org/softpat.pdf [https://perma.cc/FAW8-67GU].

argued, partially attributable to the drop in costs or the increase in cost effectiveness to obtain a software patent in the 1990s compared to the 1980s.[234]

And even under a regime of software patentability—perhaps most ironically—source code secrecy remained firmly in place. It bears mentioning that, even when software patents were being registered, the law continued to offer more solicitude to source code secrecy than one might imagine given our patent system's preference for disclosure.[235] Greg Vetter has pointed out that, just like copyright law, it was not necessary to provide source code in patent disclosure; rather, all that is needed is a description of the process implemented in the source code.[236]

2. *Narrowing the Window of Patentability*

It was only in the year 2010 that everything suddenly began to change with the onset of Federal Circuit intervention in the case of *In re Bilski*,[237] which dramatically changed the landscape for software patents. In that case, which addressed a method of hedging risk in commodity trading, the Federal Circuit explained that the claims were unpatentable on the grounds that the recited method simply comprised a computerized representation of some fundamental principles of financial risk and liability.[238] In order to satisfy the boundaries of protection, the court directed that the applicant had to either demonstrate that the claim was tied to a machine or transformed an article. In this case, however, the method was not patentable because "transformations or manipulations [of] . . . business risks[ ] or other such abstractions cannot meet the test because they are not physical objects or substances."[239]

Although the decision retained some possibility for business method protection, it explicitly pulled back on *State Street*'s standard requiring a "useful, concrete, and tangible

234 *See id.* at 9.

235 *See* Vetter, *supra* note 159, at 256 (noting that the owner may lose little from this choice).

236 *Id.*

237 545 F.3d 943 (Fed. Cir. 2008), *aff'd sub nom.* Bilski v. Kappos, 561 U.S. 593 (2010).

238 *See id.* at 963. The fact that the claimed method was performed on a computer could not transform it into something protectable, because it was basically a staple of any introductory course in finance *See id.* at 1013 (Newman, J., dissenting).

239 *See id.* at 963; *see also In re* Bilski, ELECTRONIC FRONTIER FOUND., https://www.eff.org/cases/re-bilski [https://perma.cc/64VB-R6NL].

result."[240] By the time the issue reached the Supreme Court, the Court simply upheld the rejection of the patent application on the grounds that Bilski had tried to patent an abstract idea, which was impossible under existing law.[241]

Suddenly, things had come full circle. Congress, too, began to involve itself in addressing the dubious breadth of business method patents by creating three special procedures for their review in the Leahy-Smith America Invents Act:[242] *inter partes* review, covered business method patent review, and post-grant review of issued patents.[243] Each of these procedures raised the stakes for business method patents, making it all the more likely that they could face additional challenges by others. In his discussion of the Act, Senator Leahy specifically stated that these new provisions were motivated, in no small part, by the onslaught of dubious patents that had been granted as a result of *State Street*, noting: "Patents of low quality and dubious validity, as you know, are a drag on innovation because they grant a monopoly right for an invention that should not be entitled to one under the patent law."[244]

Things further narrowed after *Alice Corp. v. CLS Bank International*[245] was handed down by the Supreme Court. Before *Alice*, in 2012, the Supreme Court had already unanimously invalidated a business method patent involving a blood diagnostic test in *Mayo Collaborative Services v. Prometheus Laboratories, Inc.*[246] The *Mayo* test added an additional wrinkle to claims that implicated laws of nature, by asking whether the claim added something more to the relevant field of analysis.[247] Then, in *Alice*, the Supreme Court returned to the issue of computer-based patents, invalidating another process for managing risks on the grounds that the patents did not amount to "significantly more" than just the abstract concept of managing risk with the use of a computer.[248] *Alice* directed the use of a two-step test to determine patentability:

> (1) whether the claim is directed to an abstract idea; and (2) if an abstract idea is present in the claim, determining whether

[240] *See In re Bilski*, 545 F.3d at 959–60 (quoting State Street Bank & Trust Co. v. Signature Fin. Grp, Inc., 149 F.3d 1368, 1373 (Fed. Cir. 1998)).

[241] *See* Bilski v. Kappos, 561 U.S. at 611.

[242] 35 U.S.C. § 102 (2011).

[243] *See* Marino & Nguyen, *supra* note 217, at 10.

[244] *Id.*

[245] 573 U.S. 208 (2014).

[246] 566 U.S. 66 (2012); *see* Marino & Nguyen, *supra* note 217, at 11–12.

[247] *See* Marino & Nguyen, *supra* note 217, at 11–12.

[248] *See id.* at 12–13.

any part of the claim amounts to significantly more than the abstract idea to qualify as an "inventive concept." If not, the claim is deemed patent ineligible.[249]

In the years after *Alice*, the Federal Circuit has largely continued to evince significant uncertainty in the field of software patents.[250] Part of the problem, commentators explain, is that both the Supreme Court and the Federal Circuit have largely failed to offer clear guidance on what comprises an abstract idea.[251] Since *Alice*, the trend has militated against protecting business method patents, with only a few exceptions.[252] One of the only Federal Circuit cases to do so, *DDR Holdings, LLC v. Hotels.com, L.P.*,[253] affirmed two patents that involved methods of generating a web page that combined certain visual elements of a host site with content from a third-party merchant on the grounds that it added enough to the abstract idea to justify patentability.[254]

In one of its clearest discussions regarding software patenting, the Federal Circuit explained in May of 2016 that it did not think that claims directed to software were inherently abstract after *Alice*, observing:

> Software can make non-abstract improvements to computer technology just as hardware improvements can, and sometimes the improvements can be accomplished through either route. We thus see no reason to conclude that all claims directed to improvements in computer-related technology, including those directed to software, are abstract and necessarily analyzed at the second step of *Alice*, nor do we believe that *Alice* so directs. Therefore, we find it relevant to ask

[249] *See id.* at 13.

[250] *See id.* at 13–19; *see also* Ultramercial, Inc. v. Hulu, LLC, 772 F.3d 709, 717 (Fed. Cir. 2014) (concluding online advertising method is not patent-eligible subject matter on abstraction grounds); Buysafe, Inc. v. Google, Inc., 765 F.3d 1350, 1355 (Fed. Cir. 2014) (concluding claims are invalid on abstraction grounds); Planet Bingo, LLC v. VKGS LLC, 576 Fed. Appx. 1005, 1008 (Fed. Cir. 2014) (affirming invalidity for a system of managing a bingo game on abstraction grounds).

[251] *See* Marino & Nguyen, *supra* note 217, at 13–19; *see also* B.J. Ard, *Notice and Remedies in Copyright Licensing*, 80 Mo. L.R. 313, 315 (2015); John Clizer, Note, *Exploring the Abstract: Patent Eligibility Post* Alice Corp. v. CLS Bank, 80 Mo. L.R. 537, 551 (2015) (noting that *Alice* did not give concrete guidance on how an abstract idea is defined).

[252] *See, e.g.*, Content Extraction & Transmission LLC v. Wells Fargo Bank, Nat'l Assoc., 776 F.3d 1343, 1348 (Fed. Cir. 2014) (noting that the patent-at-issue disclosed an abstract idea using a scanner and computer, and therefore was ineligible for protection).

[253] 773 F.3d 1245 (Fed. Cir. 2014).

[254] *See* Marino & Nguyen, *supra* note 217, at 20 (citing DDR Holdings, LLC v. Hotels.com, L.P., 773 F.3d 1245 (Fed. Cir. 2014)).

> whether the claims are directed to an improvement to computer functionality versus being directed to an abstract idea, even at the first step of the *Alice* analysis.[255]

Because the patents were not directed to an abstract idea, but instead to a specific improvement in the way that the computer operated, the patent survived.[256]

In a smattering of post-2016 cases, the Federal Circuit has remained strongly suspicious of software patents, allowing just a small window for protectability. For example, in June of 2016, the Federal Circuit found that software improvements to a filtering content tool were eligible for protection in *BASCOM Global Internet Services, Inc. v. AT&T Mobility, LLC*.[257] But then, two months later, it invalidated a system for real-time performance monitoring of an electronic power grid on the ground that it focused on independently abstract ideas that used a computer merely as a tool and was thus insufficiently inventive.[258] In short, the claims were too result-focused and functional, and they ran the risk of preempting innovation with their breadth.[259] In any event, the post-*Alice* era suggests that there is a stronger tendency to cast software patents as abstract ideas, requiring a stronger focus on whether there are additional claim elements present that can justify patentability.[260]

III
TRADE SECRECY AS DESTINATION

All of the roads I have just detailed lead back to the same place: trade secrecy as default and destination. And while this is an underlying problem from a transparency perspective, as I've argued, the roots of this problem lie in the foundational indeterminacy of software protection. In a powerful, founda-

255 *See id.* at 23 (quoting Enfish, LLC v. Microsoft Corp., 822 F.3d 1327, 1335 (Fed. Cir. 2016)).
256 *See id.* at 23–24.
257 827 F.3d 1343 (Fed. Cir. 2016); *see* Marino & Nguyen, *supra* note 217, at 23–24 (citing BASCOM Glob. Internet Servs. v. AT&T Mobility LLC, 827 F.3d 1343 (Fed. Cir. 2016)).
258 *See* Marino & Nguyen, *supra* note 217, at 24–25.
259 *See id.*
260 *See* Daniel J. Burns, *Patent Practice After* Alice, *in* DEVELOPING A PATENT STRATEGY 43, 44 (2016) ("One way to approach this analysis is to assume that software patent claims will be characterized as abstract ideas by the USPTO or by the courts and then to ask whether there are additional claim elements in the independent claims that contain an inventive concept that can transform the patent-ineligible subject matter into patent-eligible subject matter per the second part of the *Mayo* framework.").

tional article (actually, a manifesto), Pamela Samuelson, Randall Davis, Mitchell Kapor, and J.H. Reichman, two lawyers and two technologists, warned that the extension of both copyright and patent law to software might "impair the effectiveness of both forms of protection," pointing out that such overlap creates uncertainties about the scope of protection under each regime and concluding that "[n]o one knows just where the boundary line between these domains does or should lie."[261]

The real-life result of this indeterminacy is also plainly evidenced by the fact that source code remains secret at all times, irrespective of whatever regime it falls under.[262] Consider this excellent description, using a hypothetical of a person named Ariel who develops a computer program:

> Notably, Ariel does not need to publish her source code to receive protection under the intellectual property laws. She can register her program for copyright without disclosing much of the source code or executable code; rather, Copyright Office regulations require her only to disclose a portion of the code. From that portion she may even redact any trade secrets or other proprietary material. On the other hand, in order to obtain a patent, she must disclose the invention; however, such disclosure would only require a description of the invention used in the software that would enable another person working in the field to make and use the invention. It would not require her to disclose the specific code she used to implement it, or the other code that comprised the rest of the program. Thus, Ariel can receive a copyright with essentially no disclosure, and a patent with only a narrow disclosure. Moreover, if she uses trade secret law to protect the program, publication is counterproductive.[263]

In other words, as this quote demonstrates, irrespective of the changing boundaries of patent and copyright protection discussed in parts I and II, disclosure is never required, nor incentivized in any appreciable manner.

In this section, I turn toward evaluating trade secrecy on its own terms, showing how the law's own accommodation of trade secrecy in software—further cemented its underlying dominance, posing particular obstacles to the public interest in transparency.

261 Pamela Samuelson, Randall Davis, Mitchell D. Kapor & J.H. Reichman, *A Manifesto Concerning the Legal Protection of Computer Programs*, 94 COLUM. L. REV. 2308, 2346–47 (1994).

262 *See* Burns, *supra* note 260, at 44.

263 Stephen M. McJohn, *The Paradoxes of Free Software*, 9 GEO. MASON L. REV. 25, 30 (2000) (footnotes omitted).

A. The Lingering Monopoly of Trade Secrecy

Around the early 90s, scholars began to argue that copyright was the most prudent and effective area of IP to protect source code.[264] Patentability, they reasoned, was a poor fit for source code, given its lengthy duration of protection (in comparison to the short shelf life of software) and narrow subject matter.[265] And trade secret protection could essentially be claimed over much else that was kept from the public—protecting everything from disclosure, particularly whatever copyright or patent did not cover, it seemed.[266] Because much of early software was individually commissioned between a software developer and the client, the written contract became the principal way to protect against misappropriation by characterizing the software as a trade secret and requiring confidentiality.[267]

Within these practices, computer hardware companies bundled the sale of their products with software in order to optimize their hardware's capabilities and also further customized their models for the client.[268] Their client-centric business models thus enabled them to recoup their investments, Samuelson explains, without the need for copyright or patent protection.[269] And when they wanted some assurances against misappropriation, they simply characterized their source code as a trade secret and only licensed the object code to customers.

Indeed, the informal and yet ubiquitous role of trade secrecy in software beautifully illustrates its foundational justifications and tensions between them. In one influential article, Mark Lemley asserts that trade secrecy can be justified by reference to several specific areas of law—contract, tort, commercial morality, and property—and commentators and courts can vary according to their definition of which approach is the dominant one, or even if one dominates at all.[270] But is it the

264 *See* James Ryan, Comment, *The Uncertain Future: Privacy and Security in Cloud Computing*, 54 SANTA CLARA L. REV. 497, 532–33 (2014).

265 *See id.*

266 There are a great deal of articles exploring trade secrecy in software. *See, e.g.*, David Bender, *Trade Secret Protection of Software*, 38 GEO. WASH. L. REV. 909, 914 (1970) (arguing that trade secrecy provides the optimal form of protection).

267 *See* Mark A. Lemley, *Intellectual Property and Shrinkwrap Licenses*, 68 S. CAL. L. REV. 1239, 1243–45 (1995).

268 *See, e.g.*, Samuelson, *supra* note 14, at 8 n.38 (discussing IBM's use of this practice).

269 *See id.* at 8.

270 *See* Lemley, *supra* note 267, at 1270; *see also* Robert C. Scheinfeld & Gary M. Butter, *Using Trade Secret Law to Protect Computer Software*, 17 RUTGERS

nature of the property at issue that is secret? Or is it the relationship vis-à-vis the misappropriation that is at issue? At times, it is difficult to tell the difference between them, and this is especially true in the context of software.[271]

Federal law defines a trade secret to include

> information, including a formula, pattern, compilation, program, device, method, technique, or process, that: (i) derives independent economic value, actual or potential, from not being generally known to, and not being readily ascertainable by proper means by, other persons who can obtain economic value from its disclosure or use, and (ii) is the subject of efforts that are reasonable under the circumstances to maintain its secrecy.[272]

In the context of source code specifically, as the previous sections have suggested, trade secret protection extends not just to protect information that cannot satisfy the requirements of patentability or copyrightability—it extends to information that is *also* protected by those regimes as well.[273] Source code might be protected by copyright as a literary work, even though it is functional, but its functionality might also be protected by patent law through flowcharts and other representations.[274] And trade secrecy law, as Michael Risch has pointed out, rewards inventors for keeping material that is neither new nor original away from public eyes.[275]

However, without first disclosing and examining the source code, it is impossible to know whether it even qualifies as a trade secret.[276] But disclosure would potentially jeopardize its status as a trade secret. To avoid this issue, most entities

---

COMPUTER & TECH. L.J. 381, 384 (noting that some reject a property approach in favor of one that focuses on the breach of confidential trust).

[271] *See, e.g.*, E.I. Du Pont de Nemours Powder Co. v. Masland, 244 U.S. 100, 101 (1916) (considering the conflict of property rights and disclosures).

[272] *See* UNIF. TRADE SECRETS ACT § 1(4) (UNIF. LAW COMM'N 1985). There are also other federal protections in place. *See, e.g.*, Economic Espionage Act of 1996, Pub. L. No. 104-204, 110 Stat. 3488 (protecting trade secrets against theft or misappropriation in various areas such as industrial espionage); Defend Trade Secrets Act of 2016, Pub. L. No. 114-153, 130 Stat. 376 (providing a federal civil cause of action). *See* Sharon K. Sandeen & Christopher B. Seaman, *Toward a Federal Jurisprudence of Trade Secret Law*, 32 BERKELEY TECH. L.J. 829, 833 (2017) (noting the development of federal protection).

[273] *See* Samuelson, *supra* note 14, at 2–4.

[274] *See id.*

[275] *See* Michael Risch, *Why Do We Have Trade Secrets?*, 11 MARQ. INTELL. PROP. L. REV. 1, 11 (2007).

[276] *See* Charles Short, *Guilt by Machine: The Problem of Source Code Discovery in Florida DUI Prosecutions*, 61 FLA. L. REV. 177, 190 (2009) (discussing a case where the code underlying supposedly proprietary breathalyzer software was revealed to consist of nothing more than widely available, open-source code).

simply assert trade secrecy even when the underlying information may not actually qualify as a trade secret. There is no way to tell otherwise, absent some form of disclosure. It is also well settled that even a wide distribution of software programs does not compromise the intrinsic secrecy of the program as long as the program is not readily ascertainable.[277]

This strange situation, in the case of source code, produces a puzzle of inconsistency. First, consider the fact that much of source code is actually drawn from other sources, often from the public domain. Yet, because so much of the code material (i.e., the "source" of some "source code") is public in nature, the ability to keep source code from public view means that material that is closely guarded as a trade secret may not actually be secret at all.[278] Even in cases where the source code is derived from the public domain, this outcome is particularly ironic because the trade secrecy is keeping information secret that is already within the public domain. But because it is secret, we may never know this fact and never be able to challenge the source code's origins altogether.

In the past, most of the time, as James Gibson has suggested, this was completely fine with the public because the purchasing public cared not about the intricacies of the code but whether the software functions as expected.[279] The divide between public-minded protections and private controls becomes especially apparent in the software context where, as Gibson notes, "a quirk of technology allows software developers to hide from the public the very expression that earns their products copyright protection in the first place."[280]

B. Judicial Accommodation in *Kewanee*

The Supreme Court, too, has been largely untroubled by the potential rivalry between trade secrecy and patentability, and it probably never foresaw the public interest implications of this rivalry in the context of transparency.[281] For example, in the landmark case of *Kewanee Oil Co. v. Bicron Corp.*, a case that involved synthetic crystals, the Court extensively consid-

277 *See* LEMLEY ET AL., *supra* note 34, at 15.
278 *See* Risch, *supra* note 275, at 11.
279 *See* Gibson, *supra* note 20, at 175.
280 *See id.* at 171.
281 *See* Anderson, *supra* note 152, at 929.

ered the question of whether the Patent Act preempted state-protected trade secrets.[282]

Because of a seemingly clear delineation between the two areas of law, the Court concluded that the patent policy of encouraging invention was "not disturbed" by the existence of trade secrecy.[283] To justify its conclusion, the Court listed three categories of trade secrets affected by the patent regime: (1) those who were considered to be unpatentable; (2) those whose patentability was considered dubious in nature; and (3) those who were believed to qualify for patentability.

Consider the category of inventions that would be unpatentable, for example. Here, the Court reasoned, abolishing trade secret protection would not benefit disclosure in any major way because the patent alternative would be unavailable.[284] Filing doomed applications, in this instance, would not benefit disclosure to the public, it observed, because they are still kept confidential by the Patent Office.[285]

By contrast, the Court reasoned, because trade secret protection stimulates invention in areas that patent law does not cover, trade secret still encourages competition and enables the innovator to still exploit her invention.[286] Nevertheless, because trade secrecy's duration is uncertain, the Court argued that inventors would face an added push toward commercialization.[287]

But without trade secret protection, the Court reasoned, society would suffer, even in the case of unpatentable subject matter.[288] Innovative companies would be forced to engage in expensive self-help, security precautions would have to increase, and companies with limited resources would be forced to choose between the costs of added securitization or innova-

282 416 U.S. 470, 480 (1974). For an excellent analysis of Kewanee, see Sharon Sandeen, Kewanee *Revisited: Returning to First Principles to Determine the Issue of Federal Preemption*, 12 MARQ. INTELL. PROP. L.R. 299, 301 (2008).

283 *See Kewanee*, 416 U.S. at 484.

284 *See id.* at 485.

285 *Id.* Note that this rule has now changed, so that filings are now public eighteen months after filing. *See* Press Release, USPTO, USPTO Will Begin Publishing Patent Applications (Nov. 27, 2000), https://www.uspto.gov/about-us/news-updates/uspto-will-begin-publishing-patent-applications [https://perma.cc/M73J-DHRN]. For a different view, see John F. Martin, *The Myth of the 18-Month Delay in Publishing Patent Applications*, IPWATCHDOG (Aug. 3, 2015), https://www.ipwatchdog.com/2015/08/03/the-myth-of-the-18-month-delay-in-publishing-patent-applications/id=60185 [https://perma.cc/AR5G-RBXJ].

286 *See Kewanee*, 416 U.S. at 485.

287 *See id.* at 485, 494; *see also* Anderson, *supra* note 152, at 930 (stating that the uncertain duration of protection incentivizes commercialization).

288 *See Kewanee*, 416 U.S. at 484–86.

tion. Licensing and other forms of strategic discussions would level off without binding obligations of secrecy, and the public would be deprived of the benefit of the invention because fewer companies would strike agreements.

What about inventions of dubious patentability? Here, too, the Court continued to remain untroubled by the relationship between trade secrecy and patent protection. Those who have genuine doubts regarding patentability will simply opt out of the patent system, the Court predicted.[289] Others, the Court reasoned, would probably try to obtain a patent, despite the doubts, because of the comparable benefits of patent protection over trade secret protection. For those "on the line" inventors, the Court wrote, the abolition of trade secret protection would likely push them toward applying for a patent, despite the dubious outcome.[290] The nonpatentable ones will be invalidated by the Patent Office, the Court predicted. The Court explained further:

> Eliminating trade secret law for the doubtfully patentable invention is thus likely to have deleterious effects on society and patent policy which we cannot say are balanced out by the speculative gain which might result from the encouragement of some inventors with doubtfully patentable inventions which deserve patent protection to come forward and apply for patents. There is no conflict, then, between trade secret law and the patent law policy of disclosure . . . .[291]

For the final category, those that are clearly patentable, the Court noted that the disclosure value is at its peak, and the systems of trade secret versus patent protection weigh very strongly in favor of patentability. "[N]o reasonable risk of deterrence from patent application by those who can reasonably expect to be granted patents exists," the Court stated, explaining that trade secrecy provides a much weaker level of protection because it cannot bar independent inventions or reverse engineering, all of which may risk exposure and destruction of the trade secret.[292] "Where patent law acts as a barrier," it explained, "trade secret law functions relatively as a sieve."[293] In the years since, *Kewanee* has gone on to stand for a foundational presumption: trade secrecy and patent protection go hand-in-hand, and a choice between them, including the vari-

289 *See id.* at 487–88.
290 *Id.* at 488.
291 *Id.* at 489.
292 *Id.* at 489–90.
293 *Id.*

ables that go into that choice, are distinctly untroubling, often incentivizing patentability over trade secrecy.

C. Rethinking Complementarity in Software

However, there are strong reasons in place to rethink *Kewanee*'s assurances, particularly in the area of software generally. As Sharon Sandeen has argued, the Court's analysis is deeply dependent on a set of factual assumptions and its understanding about the boundaries of each area of intellectual property protection at that point in time.[294] At the time, the majority of the Court was under the impression that the availability of trade secrecy would have only a marginal effect on patent strategy because its protection seemed so much less desirable as compared to the strength of a patent grant.[295]

Today, however, things have certainly changed; as Sandeen notes, "[t]o the extent such assumptions and laws have changed, the reasoning underlying *Kewanee* must change as well or, at the very least, be re-examined."[296] Further, empirical research has shown that the reliance on trade secrecy has dramatically expanded since the 1980s, making it useful to reexamine *Kewanee*'s presumptions.[297] First, much of the opinion appears motivated by a foundational belief that trade secret law is meant primarily to protect items that might fall *outside* of the protectable boundaries of patent protection items, "which would not be proper subjects for consideration for patent protection," as the Court put it.[298]

However, the reality today is that many trade secrets might constitute otherwise patentable material.[299] Partly because of the time, effort, cost, and indeterminacy of patentability, many inventors make the rational decision to avoid patenting something when they might otherwise keep it secret. But there is

294 Sandeen, *supra* note 282, at 327.

295 Mary L. Lyndon, *Secrecy and Access in an Innovation Intensive Economy: Reordering Information Privileges in Environmental, Health, and Safety Law*, 78 U. COLO. L. REV. 465, 495 (2007).

296 Sandeen, *supra* note 282, at 327.

297 *See* Lyndon, *supra* note 295 (first citing Richard Levin et al., *Appropriating the Returns from Industrial Research and Development*, *in* BROOKINGS PAPERS ON ECON. ACTIVITY 783 (1987); then citing Wesley M. Cohen, Richard R. Nelson & John P. Walsh, *Protecting Their Intellectual Assets: Appropriability Conditions and Why U.S. Manufacturing Firms Patent (or Not)*, 12–15 (Nat'l Bureau of Econ. Research, Working Paper No. 7552, 2000) (noting the growing reliance on trade secrecy)).

298 Kewanee Oil Co. v. Bicron Corp., 416 U.S. 470, 482 (1974).

299 Michael R. McGurk & Jia W. Lu, *The Intersection of Patents and Trade Secrets*, 7 HASTINGS SCI. & TECH. L.J. 189, 199 (2015).

also another reason that pushes applicants toward secrecy: as Sandeen explains, since *Kewanee*, the law has broadened the subject matter and scope of disclosure in patent law.[300] In 1974, the only information that was disclosed to the public was an issued patent application, which required a trip to the offices of the USPTO to obtain.[301] Since 1999, the law has made all applications public eighteen months after filing, whether or not they are even issued.[302] As a result, as Sandeen notes, the end result of these developments "is that today's patent disclosure policies result in the disclosure of more information and, arguably increase innovators' interest in trade secrecy as an alternative."[303]

Second, the opinion presumes that it is easy (or even possible) for an inventor to predict *ex ante* whether the inventor will be able to obtain a patent on their invention. Especially in the case of software, most developers in a post-*Alice* world would characterize their prospects as indeterminate at least. The indeterminacy, coupled with the cost and effort of an application, actually deters rather than encourages a provisional filing, making trade secrecy that much more attractive.

Third, *Kewanee* dealt with a very different type of invention—something that was comparably more ascertainable—than the black-box source code of today. Its assurances, therefore, about the likelihood of the "ripeness-of-time concept of invention"—which suggests that others would likely reach the same solution eventually—is not always the case for software, which is sometimes heavily guarded.[304] Since source code is often outside of the public view, it makes the likelihood of such collaborative (or even comparative) innovation impossible.

These differences seriously call into question the presupposed balance between trade secrecy and patentability in the software context, justifying a need for reexamination.[305] As

300 Sandeen, *supra* note 282, at 329.

301 *Id.*

302 *Id.* at 330.

303 *Id.* In fact, Justice Marshall, in a sharply worded concurrence, disagreed strongly about the remoteness of the risk that an inventor with a patentable invention would opt for trade secret protection instead of patent protection. Because a trade secret's duration is potentially unlimited, Marshall argued that the existence of trade secret protection deprives the public of the benefit of disclosure, particularly in this case. *Kewanee*, 416 U.S. at 494–95 (Marshall, J., concurring).

304 Sandeen, *supra* note 282, at 325.

305 As an example, just consider the CONTU final report, mentioned at the opening of the predominant casebook on software:

> Although many proprietors feel secure when using trade secrecy, there are several problems they must face with respect to its use in protecting programs. Because secrecy is paramount, it is inappro-

Lemley has explained, in situations of non-self-revealing technologies, like source code (which is not evident from the sale of the product—unlike a paper clip, whose innovation is evident), secrecy can be preferable over patentability because there is more indeterminacy in the patent system.[306] As he points out, patents can be designed around, or they can be invalidated, and will eventually expire.[307] But this result produces some inefficiency because the benefits of public disclosure of the information are lost.[308] In such circumstances, it is important to note that the indeterminacy of the patent system may compel parties to opt for trade secret protection, though they might have chosen differently if the patent system were a stronger choice for protection.[309]

In those circumstances, Lemley notes, citing *Kewanee*, the defenses of independent development and reverse engineering exist to avoid a reflexive choice toward trade secrecy over patentability.[310] These defenses help make trade secrecy much less preferable, Lemley assures us as well, thereby keeping patentability within the range of attractive options.[311] While I share Lemley's views generally, I would suggest that the particular difficulties associated with reverse engineering in the software context might push the scale back toward trade secrecy.[312]

> priate for protecting works that contain the secret and are designed to be widely distributed. Although this matters little in the case of unique programs prepared for large commercial customers, it substantially precludes the use of trade secrecy with respect to programs sold in multiple copies over the counter to small businesses, schools, consumers, and hobbyists. Protection is lost when the secret is disclosed, without regard to the circumstances surrounding the disclosure. The lack of uniform national law in this area may also be perceived by proprietors as reducing the utility of this method of protection.

LEMLEY ET AL., *supra* note 34, at 5–6 (quoting CONTU, FINAL REPORT, *supra* note 112, at 34–35). Trade secrecy, the Commission noted, also reduces a company's ability to do business freely because it necessitates the signing of nondisclosure contracts. *Id.* And it also noted that the reduced flow of information due to secrecy reduces the consumers' ability to comparison shop, leading to higher prices. *Id.*

306 LEMLEY ET AL., *supra* note 34, at 339–40.

307 *Id.* at 340.

308 *Id.*

309 *Id.*

310 *Id.* at 340–41.

311 *Id.*

312 A similar comment on Lemley is offered by Jeanne Fromer, who points out that there is evidence from some industries that innovators will still take excessive precautions to protect their secrets because the legal remedies for misappropriation are often incomparable to the losses faced from the extinguishing of a secret. Fromer, *supra* note 190, at 15–16.

Closed code also carries deleterious impacts for software innovation. Not only does the public lose more of the public domain, but other developers are unable to build on others' innovations, making it impossible to optimize efficiency or increase interoperability without licensing the code first.[313] The CONTU report, for example, noted that humans waste a lot of effort trying to create what is already held in secret.[314]

Because the code is unavailable to anyone outside of the company, third parties who might seek to improve upon the code are unable to do so without permission, stymying the development of markets for innovation.[315] Since trade secret laws encourage designers to "build fences" around their secrets, information is often only sparingly revealed, and then only under stringent conditions of nondisclosure.[316] These conditions intrinsically discourage the sharing of information, impeding market-wide vertical interoperability.[317] As Jeanne Fromer has observed, the failure to share this information with the wider public contributes to an information asymmetry between the initial innovator and the follow-on competitor, reducing the democratization of innovation.[318] Moreover, the trend toward secrecy also means that a developer may not actually detect infringement because a programmer may find themselves stymied from proving piracy without expending considerable resources to obtain discovery.[319]

Further, a trade secrecy regime not only makes it impossible to compare works with those that exist in the public domain, it also shrinks the size of the public domain altogether.[320] Fair use may be a laudable public right of access, but it is meaningless in the face of access restrictions that deny entrance to all unlicensed uses, fair and nonfair alike.[321] While some courts have used the fair use doctrine to protect temporary, technically infringing behavior, like copying code or

[313] Gibson, *supra* note 20, at 181.

[314] LEMLEY ET AL., *supra* note 34, at 6 (citing CONTU, FINAL REPORT, *supra* note 112, at 34–35).

[315] Gibson, *supra* note 20, at 184.

[316] Lemley & O'Brien, *supra* note 125, at 290–91.

[317] *Id.*

[318] *See* Fromer, *supra* note 190, at 14.

[319] Gibson, *supra* note 20, at 187.

[320] *Id.* at 183; Strandburg, *supra* note 150, at 105–06.

[321] Gibson, *supra* note 20, at 171. As an example of this complexity, consider the longstanding litigation in the Google/Oracle fair use case. *See* Peter S. Menell, *API Copyrightability Bleak House: Unraveling and Repairing the* Oracle v. Google *Jurisdictional Mess*, 31 BERKELEY TECH. L.J. 1515, 1521–62 (2016) (discussing the many stages of litigation).

copyrightable material if it is the only way to access material in the public domain, the doctrine is limited by its imprecision.[322] Since most source code remains unpublished, it becomes harder to avail oneself of fair use protections in that context.[323] Not only are unpublished works subject to greater protection than published works, but since fair use is usually considered only a defense, it does not provide the means to actually access closed code.[324]

As a result of these shortcomings of intellectual property protection to incentivize disclosure and access, source code remains entirely secluded from outside view, maximizing the developer's control, irrespective of whether the goals of third party access lie in innovation, competition, or investigation.

IV
DUE PROCESS IN AN AGE OF DELEGATION

The ubiquity of trade secrecy in the arena of source code, as I suggested above, has dramatic implications for innovation, interoperability, and competition. Although those implications can be deleterious in the context of private industry, more troubling is the implications of closed code on the functions of public governance.[325] Danielle Citron, ten years ago, observed that the administrative state was slowly being overtaken by closed-proprietary systems in areas of public benefits, electronic voting, and agency-gathered data, among others.[326] Today, the issue is not just that government decision making is becoming entirely privatized, it is also that these systems are closed and proprietary, often due to assertions of trade secrecy. David Levine offers several examples—from telecommunications to traditional government operations, like voting—that are now being provided by private industry and immunized from transparency by trade secret doctrine.[327] Particularly in the realm of public infrastructure, secrecy has skyrocketed in importance—one study cited by Levine mentions that in

[322] Gibson, *supra* note 20, at 192.
[323] *Id.*
[324] *Id.* at 193.
[325] Citron, *supra* note 8, at 363–71.
[326] *Id.*
[327] David S. Levine, *The Impact of Trade Secrecy on Public Transparency*, *in* THE LAW AND THEORY OF TRADE SECRECY: A HANDBOOK OF CONTEMPORARY RESEARCH, *supra* note 83, at 406, 407.

twenty-four out of thirty-three manufacturing industries, secrecy was ranked as first or second in importance.[328]

Although the risks of privatization are not at all new to legal scholarship,[329] few scholars have linked the rise of privatization to the reliance on closed code, automated governance, and the rise of trade secrecy. As I suggested in an earlier article, we continue to view trade secrecy as somehow separate from civil rights concerns, and that presumption has facilitated the absence of accountability.[330]

In this section, I first discuss the rise of delegation to private industries and the range of trade secrecy claims that have pervaded attempts toward transparency. In the second section, I discuss some of the ways in which similar issues of privatization and delegation have emerged in source code disputes in the criminal context, and the implications of those decisions on the liberty and due process interests of criminal defendants. Finally, in the last section, I discuss the increased reliance on trade secrecy and the Computer Fraud and Abuse Act to preclude attempts toward greater transparency and disclosure.

A. The Rise of Closed Code Governance

As the story that opened this Article demonstrates, many municipalities are confronting the implications of enabling private industry, instead of the government, to make decisions about the lives and services provided to citizens.[331] When automated decision making and trade secrecy facilitates this intermingling of public and private, it produces a crisis of transparency. In this context, private businesses now play the roles that government used to play but can utilize the principles of trade secret law to insulate themselves from the very

[328] *Id.* at 408 (citing Gerald Carlino et al., *Matching and Learning in Cities: Urban Density and the Rate of Invention* 5 (Fed. Reserve Bank of Phila., Working Paper No. 04-16, 2006)).

[329] *See* Craig Anthony (Tony) Arnold, *Privatization of Public Water Services: The States' Role in Ensuring Public Accountability*, 32 PEPP. L. REV. 561, 562 (2005); Laura A. Dickinson, *Public Law Values in a Privatized World*, 31 YALE J. INT'L L. 383, 384–85 (2006) (privatization of military support services); Matthew Diller, *Form and Substance in the Privatization of Poverty Programs*, 49 UCLA L. REV. 1739, 1739 (2002); Martha Minow, *Public and Private Partnerships: Accounting for the New Religion*, 116 HARV. L. REV. 1229, 1230–31 (2003) (school privatization efforts); David E. Pozen, *Managing a Correctional Marketplace: Prison Privatization in the United States and the United Kingdom*, 19 J.L. & POL. 253, 253 (2003) (privatization of prisons).

[330] *See* Katyal, *supra* note 17, at 118.

[331] McKenzie, *supra* note 1.

expectations of accountability under which that government operated.[332]

These tensions—between democratic transparency and commercial seclusion—have become particularly pronounced in the current day, where government has become increasingly intermingled with private industry through privatization and delegation throughout infrastructure involving telecommunications, government operations, and energy.[333] As Gillian Metzger has observed, "[p]rivatization is now virtually a national obsession."[334] Her work describes privatization in the context of the sharing of responsibility between public and private but with a twist: instead of the government ensuring control over its programs, the private industry takes the lead.[335] In an exhaustive account, Metzger describes the expansion of privatization in areas like Medicare, Medicaid, welfare programs, public education, and prisons.[336] In each of these contexts, private contractors exercise a broad level of authority over their program participants, even when government officials continue to make determinations of basic eligibility and other major decisions.[337]

While Metzger's focus is on the privatization of government services, each involving a delegation to a private entity, I would underscore that much of the privatization that she studies is also facilitated by an additional focus on automated decision making. As Robert Brauneis and Ellen Goodman have elo-

332 *See* Katyal, *supra* note 17, at 118–19; Levine, *supra* note 327, at 2.

333 Levine, *supra* note 327, at 2; David S. Levine, *Secrecy and Unaccountability: Trade Secrets in Our Public Infrastructure*, 59 FLA. L. REV. 135, 135 (2007).

334 Gillian E. Metzger, *Privatization as Delegation*, 103 COLUM. L. REV. 1367, 1369 (2003); *see also* Alfred C. Aman, Jr., *Globalization, Democracy, and the Need for a New Administrative Law*, 49 UCLA L. REV. 1687, 1700–03 (2002) (discussing democracy issues raised by privatization of prisons and social services for the poor); Matthew Diller, *Going Private—the Future of Social Welfare Policy?*, 35 CLEARINGHOUSE REV. 491, 491 (2001) (discussing "broad movement to 'privatize' government [poverty] programs"); Mathew Diller, *Introduction: Redefining the Public Sector: Accountability and Democracy in the Era of Privatization*, 28 FORDHAM URB. L.J. 1307, 1308 (2001) (describing privatization of government services, including "contracting out the delivery of services, divestiture of government owned resources and institutions, the establishment of private communities with quasi-governmental powers, the creation of voucher programs to replace the direct delivery of services, the movement toward incentive-based or private forms of regulation, and the possible replacement of the Social Security system with individual savings accounts"); Mark H. Moore, *Introduction*, 116 HARV. L. REV. 1212, 1212 (2003) (introducing a symposium "focus[ed] on the increased 'privatization' of the public sphere").

335 Metzger, *supra* note 334, at 1370.

336 *Id.* at 1380.

337 *Id.* at 1387.

quently noted, "[t]he risk is that the opacity of the algorithm enables corporate capture of public power."[338] There are also secondary, less visible forms of automated decision making that can also amount to a significant, though related, degree of delegation to private entities involving contracting with private entities for the purposes of information gathering or distribution.

In this context, the government can and has asserted its own trade secret protection as an exemption to disclosure under the Freedom of Information Act.[339] David Levine has documented a number of situations where the government has claimed trade secrecy in a wide variety of scenarios, including situations where a government entity is directly competing with a private sector entity or acting as a provider of particular goods, and where the government has contracted with a private entity.[340] Examples involve government record keeping, government-run student loan assistance,[341] and even a government firearm registry.[342] In another case, Cincinnati Public Schools maintained that their ninth-grade multiple choice and essay questions were protected trade secrets.[343] In yet another, the United States Air Force maintained that details regarding pricing and particular options on a private contract with McDonnell Douglas Corporation were protected trade secrets free from public transparency.[344]

In the context of private firms, the issue of opacity deepens even further. Here, firms have learned to obfuscate trans-

338 *See* Robert Brauneis & Ellen P. Goodman, *Algorithmic Transparency for the Smart City*, 20 YALE J.L. & TECH. 103, 109 (2018); *see also* Will Knight, *The Dark Secret at the Heart of AI*, MIT TECH. REV. (Apr. 11, 2017), https://www.technologyreview.com/s/604087/the-dark-secret-at-the-heart-of-ai/ [https://perma.cc/PA6F-HYDT] ("No one really knows how the most advanced algorithms do what they do.").

339 *See* David S. Levine, *The People's Trade Secrets?*, 18 MICH. TELECOMM. & TECH. L. REV. 61, 82 (2011).

340 *See generally infra* pt. II; Levine, *supra* note 339, at 84 (discussing situations in which governments have asserted trade secrecy).

341 Levine, *supra* note 339, at 82 (citing Pelto v. Connecticut, No. FIC 2008-341 ¶ 32 (Conn. Freedom of Info. Comm'n May 13, 2009) (final decision), https://www.state.ct.us/foi/2009FD/20090513/FIC2008-341.htm [https://perma.cc/U2C6-CWJA]); *see also* Hoffman v. Pennsylvania, 455 A.2d 731, 733 (Pa. Commw. Ct. 1983) (where a plaintiff sought magazine subscriber mailing lists).

342 *See* Levine, *supra* note 339, at 90; *see also* OFF. OF THE INFO. COMM'R OF CAN., ANNUAL REPORT INFORMATION COMMISSIONER 1999-2000, at 60 (2000) (explaining the Canadian government's firearm registry).

343 Levine, *supra* note 339, at 83; *see* State *ex rel.* Perrea v. Cincinnati Pub. Schs., 916 N.E.2d 1049, 1052–53 (Ohio 2009).

344 Levine, *supra* note 339, at 99; *see* McDonnell Douglas Corp. v. Widnall, 57 F.3d 1162, 1163–64 (D.C. Cir. 1995).

parency by relying on assertions of trade secrecy to avoid disclosure of data in the context of environmental, health, and safety data.[345] Even when disclosures are mandated by regulation, Mary Lyndon has argued that nondisclosure privileges have grown, leading to trends that tend to favor commercial interests over public ones.[346] The issue of trade secrecy impeding the public interest has come up in a variety of disputes, including health care devices and clinical trials, voting machines, breathalyzer disputes, and search-engine algorithms.[347] Particularly in the context of health or environmental concerns, which are often underestimated because they are not immediately visible, firms may resist disclosing information on the grounds that it may disadvantage them commercially.[348] Annemarie Bridy has shown how medical device manufacturers have attempted to keep their pricing information secret as a way to keep information away from their customers.[349] In another environmental context, after chemicals leaked from a West Virginia coal processing plant, denying over 300,000 people access to water, the plant successfully refused to turn over the specific makeup of its compounds to the public.[350]

While the sheer variety of these instances deserves a more comprehensive and searching investigation in the context of AI,[351] these cases suggest two notable elements, each linked to one another. The first involves the element of privatization, exemplified by the existence of a contractual relationship with a private party. The second element is one of (what I call) "information insulation," involving an increased willingness to assert trade secret protection in cases where transparency

345 *See* Lyndon, *supra* note 295, at 471.

346 *Id.* at 509.

347 *See* Deepa Varadarajan, *Trade Secret Fair Use*, 83 FORDHAM L. REV. 1401, 1443–44 (2014); Lyndon, *supra* note 295; *see also* FRANK PASQUALE, THE BLACK BOX SOCIETY: THE SECRET ALGORITHMS THAT CONTROL MONEY AND INFORMATION (2015) 140–44; Rebecca S. Eisenberg, *Data Secrecy in the Age of Regulatory Exclusivity*, *in* THE LAW AND THEORY OF TRADE SECRECY: A HANDBOOK OF CONTEMPORARY RESEARCH, *supra* note 83, at 467, 470.

348 Lyndon, *supra* note 295.

349 See Annemarie Bridy, *Trade Secret Prices and High-Tech Devices: How Medical Device Manufacturers Are Seeking to Sustain Profits by Propertizing Prices*, 17 TEX. INTELL. PROP. L.J. 187, 191 (2009), which is discussed in Varadarajan, *supra* note 347, at 1442–43.

350 Varadarajan, *supra* note 347, at 1443.

351 In future work, I plan to investigate these cases and others. *See generally* Sonia K. Katyal, Delegated Decision Making and Government Transparency (abstract) (on file with author).

might be justified due to public interest concerns.[352] While Metzger focuses on the dangers of narrowing state action in such contexts, highlighting the vagaries of discretionary decision making, Levine and others emphasize, troublingly, how these powers can become even more insulated from the public eye through protections from disclosure to the public. Added to these risks today is the even greater power of government-sponsored automated decision making, amplifying even further the risks to government accountability and due process.

The risks to accountability and transparency affect both individualized cases as well as our democratic system. But they become particularly pronounced in cases involving source code secrecy. Consider an example. In more than one voting issue, assertions of trade secrecy prevented election officials from releasing software to independent auditors to enable review and testing.[353] In 2005, a voting machine company, Diebold Election Systems (now called Premier Election Solutions), refused to follow a North Carolina law that required electronic voting machine manufacturers to place its software and source code in escrow with a state Board of Elections-approved agent.[354] Over a series of court battles, Diebold refused to comply, eventually withdrawing from the state altogether, rather than reveal its source code.[355] In another event, also discussed by Levine, when hackers successfully accessed (and manipulated) a series of Diebold machines, Diebold chose to characterize the events as "potential violations of licensing agreements and intellectual property rights," rather than re-

[352] *See* Levine, *supra* note 339, at 111 (discussing the public interest concerns at stake).

[353] *See* Andrew Massey, *"But We Have to Protect our Source!": How Electronic Voting Companies' Proprietary Code Ruins Elections*, 27 HASTINGS COMM. & ENT. L.J. 233, 235 (2004); Brenda Reddix-Smalls, *Individual Liberties and Intellectual Property Protection—Proprietary Software in Digital Electronic Voting Machines: The Clash Between a Private Right and a Public Good in an Oligopolistic Market*, 19 FORDHAM INTELL. PROP. MEDIA & ENT. L.J. 689, 742–43 (2009).

[354] Levine, *supra* note 327, at 96. For an excellent article exploring the use of software-independent voting systems, compliance audits, and risk-limiting audits in elections, see P.B. Stark & D.A. Wagner, *Evidence-Based Elections*, IEEE SECURITY & PRIVACY, Sept. 2012, at 33 (spec. issue on electronic voting).

[355] Michael A. Carrier, *Vote Counting, Technology, and Unintended Consequences*, 79 ST. JOHN'S L. REV. 645, 667–68 (2005); Levine, *supra* note 327, at 13; Doris Estelle Long, *"Electronic Voting Rights and the DMCA: Another Blast from the Digital Pirates or a Final Wake Up Call for Reform?"*, 23 J. MARSHALL J. COMPUTER & INFO. L. 533, 545–48 (2005).

spond to the threat to the democratic dignity of the voting tabulation process.[356]

B. The Constitutional Cost of Secrecy

As I have suggested above, one of the primary obstacles to greater transparency involves the increasing privatization of government functions. While the prior section addressed this issue in the context of trade secrecy, it is also important to understand the significance of this difference in the context of comparing how private, data-driven decisions are often free from scrutiny, as compared to decisions made directly by the state.

Nowhere is this becoming more apparent than in the context of criminal law. In the criminal law context, computer-processing technologies have been employed in criminal prosecutions involving fingerprinting, DNA match analysis, facial recognition, drunk driving, and file sharing.[357] A further complexity within criminal law lies in the use of Automated Suspicion Algorithms (ASAs), which apply machine learning to data with the purpose of identifying individuals who may be engaged in criminal activity and may produce conflicts with the Supreme Court requirement of individualized suspicion under the Fourth Amendment.[358] In an eloquent and comprehensive article, Rebecca Wexler examines a host of these automated decision-making procedures in the life cycle of a criminal justice case, including bail investigations, trial evidence, sentencing, and parole, noting the substantial deference that courts have extended to trade secret owners in every one of these areas, even though their processes (and the decisions that they reach) often implicate the difference between liberty and imprisonment.[359]

Issues of admissibility and reliability further highlight the contradictory paradox of source code secrecy: on one hand, companies argue that their methods are sufficiently known and proven to be broadly accepted by the scientific community and yet, on the other hand, companies will go to enormous lengths

[356] Levine, *Secrecy and Unaccountability*, *supra* note 333, at 182 (quoting Leon County Supervisor of Elections Ion Sancho: "I really think they're not engaged in this discussion of how to make elections safer.").

[357] Chessman, *supra* note 43, at 180–81.

[358] *See* Michael L. Rich, *Machine Learning, Automated Suspicion Algorithms, and the Fourth Amendment*, 164 U. PA. L. REV. 871, 886 (2016) (discussing ASAs and individualized suspicion).

[359] *See* Wexler, *Life, Liberty, and Trade Secrets*, *supra* note 12, at 9.

to keep their source code confidential so as to preclude further investigation.[360]

Assertions of trade secret privilege in most states are covered by sections of the evidence code, which provides for protection from disclosure as long as it will not "conceal fraud or otherwise work injustice."[361] Courts have interpreted this provision to also include a requirement that the defense in a criminal case must also show that the trade secret is relevant and necessary to the defense in order to obtain disclosure under a protective order.[362] In one criminal case involving DNA analysis and its TrueAllele program, Cybergenetics maintained that it kept the source code secret because of the "highly competitive commercial environment," and it provided defense experts with its methodology and underlying mathematical model, arguing that its source code was unnecessary to assess the program's reliability.[363] The court agreed with Cybergenetics, concluding that its source code was not necessary to determine the software's reliability and that the defense had failed to demonstrate a particularized showing of need.[364] It further rejected the prospect of a Sixth Amendment violation, holding that the Confrontation Clause did not require pretrial discovery of privileged information.[365] This outcome is hardly an anomaly.[366]

Yet, according to experts, TrueAllele's match statistic values dramatically diverge from the findings of other competitors.[367] Whereas other competitors found DNA analysis to be

---

360 *See* Katherine L. Moss, Note, *The Admissibility of TrueAllele: A Computerized DNA Interpretation System*, 72 WASH. & LEE L. REV. 1033, 1071–72 (2015); *see also* William C. Thompson & Simon Ford, *DNA Typing: Acceptance and Weight of the New Genetic Identification Tests*, 75 VA. L. REV. 45, 59–60 (1989) (noting that asserting trade secrecy shields companies from scrutiny by the scientific community); Stephanie L. Damon-Moore, Note, *Trial Judges and the Forensic Science Problem*, 92 N.Y.U. L. REV. 1532, 1536 (2017) (discussing "constraints on judges' abilities to recognize and address problems with forensic science").

361 *See, e.g.*, CAL. EVID. CODE § 1060 (West 2018); People v. Chubbs, No. B258569, 2015 WL 139069, at *10–14 (Cal. Ct. App. Jan. 09, 2015).

362 *See Chubbs*, 2015 WL 139069, at *6–7.

363 *Id.* at *8.

364 *Id.* at *10; *see also* Commonwealth v. Foley, 38 A.3d 882, 889–90 (Pa. Super. Ct. 2012) (reaching the same conclusion).

365 *See Chubbs*, 2015 WL 139069, at *11.

366 Several other courts have reached similar conclusions on TrueAllele. *See Foley*, 38 A.3d at 890; *see also* Moss, *supra* note 360, at 1062–68 (citing cases).

367 *See* Brief of the Innocence Project, Inc. et al. as Amici Curiae Supporting Respondents at 13, People v. Johnson, No. F071640 (Cal. Ct. App. 2019); *see also* Chessman, *supra* note 43, at 198 (discussing how widely a RMP calculated by TrueAllele diverged from a RMP calculated by a conventional DNA lab using the same data).

too unreliable, TrueAllele offered "match statistics of astounding confidence."[368] At the same time, TrueAllele's repeat analyses have reached different outcomes with the same data, raising concerns about admissibility due to these internal inconsistencies.[369] Under these circumstances, it is virtually impossible to detect errors. And errors can often mean the difference between liberty and imprisonment. For example, consider that source code errors in other genotyping software programs, like STRmix, produced materially-altered match statistics in over sixty cases.[370] It is precisely to address that problem that STRmix now provides access to its source code when it is used to generate evidence in prosecutions.[371] Notably, it is also the key reason why, in September 2016, New York City decided to retire Forensic Statistical Tool, a previous in-house tool, in favor of STRmix.[372]

I have discussed the risks of privatization in a variety of contexts, but consider another set of scenarios that illustrate its implications. In one criminal case, a defendant was unable to acquire the source code to challenge his breath-alcohol score for a simple but surprising reason.[373] Since discovery orders are limited to items or information within the custody, possession, or control by the State, and since the source code was held by the manufacturer and considered to be a trade secret, the court refused to require it to be turned over because it was essentially out of the boundaries of the discovery order.[374] At least eight states have denied defendants access to source code due to similar issues of trade secrecy.[375] In some cases, states will argue that they lack possession of the source code and therefore cannot turn it over for investigation.[376] And the court will adopt this rationale even to the detriment of the defendant. In at least one case, in order to assist prosecutors, law enforcement deliberately avoided taking possession of the source code

368 Brief of the Innocence Project, *Johnson*, No. F071640, at 12.
369 *Id.* at 13.
370 *Id.* at 18–19 (citing David Murray, *Queensland Authorities Confirm 'Miscode' Affects DNA Evidence in Criminal Cases*, COURIER MAIL (Mar. 20, 2015), https://www.couriermail.com.au/news/queensland/queensland-authorities-confirm-miscode-affects-dna-evidence-in-criminal-cases/news-story/833c580d3f1c59039efd1a2ef55af92b [https://perma.cc/YCR7-ZLZW]).
371 *See id.* at 19 (citing ESR, ACCESS TO STRMIX SOFTWARE BY DEFENCE LEGAL TEAMS (2016), https://strmix.esr.cri.nz/assets/Uploads/Defence-Access-to-STRmix-April-2016.pdf [https://perma.cc/BQF3-9TBP]).
372 *Id.*
373 State v. Kuhl, 741 N.W.2d 701, 708 (Neb. Ct. App. 2007).
374 *Id.*
375 Wexler, *Life, Liberty, and Trade Secrets*, *supra* note 12, at 7.
376 Chessman, *supra* note 43, at 213–14.

in order to avoid turning the code over to defense counsel or the defense's expert.[377]

Imagine the effect of such a finding on the landscape of constitutional or human rights—it would essentially mean that every time the state handed over information to a private party that then asserted trade secret protection, it would be out of the bounds of discovery unless the party was willing to seek a subpoena. Effectively, these cases suggest that through assertions of trade secrecy, the state is practically able to immunize itself from investigation regarding its forensic techniques. In other criminal cases, defendants have lost because courts would reject the proposition that access to the source code was necessary for a defense. Wexler details the case of a California appeals court that upheld a software developer's refusal to comply with a trial court order to turn over the source code for a forensic software program used to convict the defendant on the grounds that the code was not relevant or necessary to the defense.[378] Similar refusals to compel source code have occurred in the context of the Intoxilyzer, which is used to measure alcohol intoxication.[379] In a similar context involving Alcotest, a popular breath test device, the company refused to sell its device to non-law enforcement entities to enable independent verification on trade secrecy grounds.[380]

More troublingly, consider the lines between privatization and public responsibilities. Here, the private status of the manufacturer facilitates the striking dismissal of core constitutional protections regarding the right to confront witnesses at trial. However, as Christian Chessman observed, there is an even greater irony operating here.[381] In these decisions, both state and federal courts routinely presume the reliability and accuracy of the techniques they rely upon.[382] And yet, computer scientists would argue exactly the reverse: that the programs themselves do not automatically or inherently ensure reliability.[383] As Chessman writes, "computer programs are not more reliable than human statements because they *are*

377 *Id.*

378 Wexler, *Life, Liberty, and Trade Secrets*, *supra* note 12, at 7 (discussing People v. Chubbs, No. B258569, 2015 WL 139069 (Cal. Ct. App. Jan. 9, 2015)).

379 *See* Natalie Ram, *Innovating Criminal Justice*, 112 NW. U. L. REV. 659, 662 (2018).

380 *Id.* at 672 (citing State v. Chun, 943 A.2d 114 (N.J. 2008)).

381 Chessman, *supra* note 43, at 183.

382 *Id.* at 184.

383 *Id.*

human statements—and no more than human statements."[384] Since they are tools of human design, they are often subject to human error, faulty assumptions, and mistakes, just like any other kind of evidentiary tool.[385] This is perhaps the strongest reason for why machine testimony deserves the benefit of adversarially-generated scrutiny.[386] Errors can constantly reproduce because each program update can interact negatively with preexisting code.[387]

These issues are by no means limited to the government. In the context of scientific research, academics often offer general conceptual and functional descriptions of scientific-created software and withhold source code in favor of releasing only the binary, executable version.[388] This affects the process of peer review, making it impossible to detect errors from reproducing results, leading some to allege that the disclosure problem has led to a "credibility crisis" in research computation.[389]

C. The New Secrecy: Information Insulation

As the previous sections have demonstrated, source code secrecy can have dramatic implications for the public interest, particularly in the area of criminal justice. Here, rather than recognizing the deep complexity of trade secret law (and its limitations), courts are tending to defer to trade secret owners, often to the detriment of the public interest.[390]

Today, the circumstances under which trade secrecy is asserted, I would argue, change the traditional function of trade secrecy from protecting against a competitor's misappropriation to a function that impedes public investigation. Early trade secret cases raise paradigmatic fact patterns that involve some form of misappropriation: circumstances where departing employees sought to continue their business; or competitors copied another's products; or contracts to keep certain

384 *Id.* at 186.

385 *Id.* at 184.

386 Andrea Roth, *Machine Testimony*, 126 YALE L.J. 1972, 1976–78 (2017).

387 Chessman, *supra* note 43, at 185.

388 *See* Darrel C. Ince et al., *The Case for Open Computer Programs*, 482 NATURE 485, 486–87 (2012) (expressing concern about the need to share source code among scientific researchers); A. Morin et al., *Shining Light into Black Boxes*, 336 SCI. 159, 159 (2012) (expressing the same concern).

389 Morin et al., *supra* note 388, at 160. In 2010, of the twenty most-cited science journals, only three had policies requiring source code disclosure, in contrast to near-universal agreement requiring the availability of other forms of data. *Id.* at 161.

390 I am grateful to Tait Graves for this helpful observation.

business information confidential.[391] In one of the earliest descriptions of trade secrets, the Supreme Court of Massachusetts observed in 1868:

> If [a person] invents or discovers, and keeps secret, a process of manufacture, whether a proper subject for a patent or not, he has not indeed an exclusive right to it as against the public, or against those who in good faith acquire knowledge of it; but he has a property in it, which a court of chancery will protect against one who in violation of contract and breach of confidence undertakes to apply it to his own use, or to disclose it to third persons.[392]

Hundreds of years later, this summary still applies to most cases of trade secrecy.[393] The typical defendant in trade secrecy cases involves a competitor who has allegedly misappropriated the plaintiff's trade secret for profit and unfair competition.[394]

Yet, more recently, the circumstances I discuss in this Article demonstrate three core differences from the classic cases involving trade secrecy. First, in all of the examples we have examined here, the defendant's motivation is not to compete with a trade secret holder but rather to investigate a particular source of information. Here, the concern is not motivated by misappropriation for the purposes of competition, but rather for the purposes of discovery or investigation. Second, unlike the classic trade secrecy cases, the parties that are usually at odds with one another have no formal, preexisting contractual relationship—the source code is sought for the purposes of disclosure to the public or for the purposes of investigation of bias, not for the purposes of financial gain. Third, in many of these examples, the government plays some key role, either because it is prosecuting the case or because it is acting in a decision-making capacity.

All of these differences, I think, help to underscore the role of trade secrecy as an obstacle to the public interest. But it requires us to think differently about how to address the role of

---

391 *See* Lemley, *supra* note 176, at 315.

392 *See* Risch, *supra* note 275, at 13 (quoting Peabody v. Norfolk, 98 Mass. 452, 458 (1868)).

393 *See* Charles Tait Graves & Brian D. Range, *Identification of Trade Secret Claims in Litigation: Solutions for a Ubiquitous Dispute*, 5 NW. J. TECH & INTELL. PROP. 68, 72 (2006) ("A trade secret case usually begins shortly after a former employee has resigned and either joined a competitor or formed a new, competing business.").

394 *See, e.g.*, DVD Copy Control Ass'n v. Bunner, 10 Cal. Rptr. 3d 185, 195 (Cal. Ct. App. 2004) (involving such a claim); *see also* Risch, *supra* note 275, at 15 (noting that this may be the modern view of trade secrets litigation).

trade secrecy in these cases of information insulation. As the importance of trade secrecy has increased, so has surrounding litigation, which has grown exponentially since the 1980s.[395]

And as litigation has increased, in the civil context, so have the attempts to insulate trade secrets from inquiry and investigation. As two leading trade secret experts have explained, it is typical for the plaintiff to avoid a specific identification of the trade secret precisely to obfuscate inquiry.[396] Instead, the plaintiff argues "that the defendant already knows what the alleged trade secrets are because the defendant knows what it stole, and thus no identification is necessary."[397] In these cases, the plaintiff will rarely provide a precise and complete identification of the trade secrets unless a court forces them to do so.[398]

If the trade secret owner avoids identifying its trade secrets in a classic departing-employee case on the grounds of familiarity, imagine how much more difficult it can be to obtain the information when the interest at stake involves allegations of bias. Such cases do not involve misappropriation for the purposes of unfair competition, but they implicate core concerns about fairness and accountability to the public. These interests would only escalate the plaintiff's impetus to avoid discovery and identification.

Three results flow from this observation. First, assertions of trade secret protection, just as the prior section suggests, remain a key obstacle for researchers and litigants seeking to test the efficacy and fairness of government algorithms and automated decision making.[399] Even the most effective investigations, like ProPublica's projects, which have addressed a myriad number of issues (Uber's surge pricing, Amazon's pricing algorithm, and the COMPAS recidivism algorithm, among others), have been undertaken without access to the underly-

395 *See* David S. Almeling, Darin W. Snyder, Michael Sapoznikow, Whitney E. McCollum & Jill Weader, *A Statistical Analysis of Trade Secret Litigation in Federal Courts*, 45 GONZ. L. REV. 291, 293 (2009).

396 Graves & Range, *supra* note 393, at 72.

397 *Id.*

398 *Id.* at 68.

399 Christian Sandvig et al., *Auditing Algorithms: Research Methods for Detecting Discrimination on Internet Platforms* 9 (Paper Presented to Data and Discrimination: Converting Critical Concerns into Productive Inquiry, 64th Annual Meeting of the Int'l Commc'n Assoc. Preconference, May 22, 2014, https://www-personal.umich.edu/~csandvig/research/Auditing%20Algorithms%20--%20Sandvig%20--%20ICA%202014%20Data%20and%20Discrimination%20Preconference.pdf [https://perma.cc/HF8J-BX6X].

ing source code, forcing investigators to perform audits without access to key data.

Second, the conventional exceptions to trade secret protection within the law—reverse engineering, for example—are usually unavailable in the context of AI. If the source code is unavailable, the only way to obtain the code is to engage in reverse engineering, but this is often difficult, costly, and restricted, either by copyright law (which prohibits reverse engineering for the purposes of copying or duplication) or by contract.[400] Michael Mattioli has argued, "unlike software, big data practices cannot be reverse engineered. That is, an expert cannot decipher just how a set of data was assembled with nothing more to work from than the data itself."[401] Because the computer code for an algorithm is so complex, simply reading the code does not make it interpretable without the ability to plug in data and see how the algorithm actually functions.[402] In addition, because algorithms increasingly depend on the input of unique personal data, the outcomes may be obscure and difficult to study in a systematic capacity without access to the data.[403] Finally, there are other issues raised from relying on self-reporting data as well.[404]

Last, legal threats have stymied attempts toward investigation and transparency. Consider this example. In 2005, an employee of Internet Security Systems, Michael Lynn, was asked to reverse engineer Cisco's Internet Operating System (IOS), which served as the operating system for Cisco's routers used by both private and public entities.[405] Lynn discovered that the system had a security vulnerability, known as "exploit

400 *See Source Code Definition*, *supra* note 45, at 3.

401 *See* Michael Mattioli, *Disclosing Big Data*, 99 MINN. L. REV. 535, 550–53 (2015) (citing Peter S. Menell, *The Challenges of Reforming Intellectual Property Protection for Computer Software*, 94 COLUM. L. REV. 2644, 2652 (1994) (noting the use of trade-secret protection in software industry)).

402 Sandvig et al., *supra* note 399, at 10.

403 *Id.* (noting that the input of unique personal data means that "the same programmatically-generated Web page may never be generated twice"). It is also difficult to investigate when the data itself is proprietary, which is often the case. *See generally* Amanda Levendowski, *How Copyright Law Can Fix Artificial Intelligence's Implicit Bias Problem*, 93 WASH. L.R. 579, 605, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3024938 [https://perma.cc/P93M-RX35] (discussing how copyright law, which restricts access to training data, limits algorithmic accountability, including transparency).

404 Noninvasive user audits, which involves sharing the search queries from users and their results (with their consent), have the advantage of not disturbing the platform itself but can result in a serious sampling issue if the users queried are not representative of the entire database, and so run the risk of reproducing other kinds of errors. *Id.* at 11.

405 Levine, *Secrecy and Unaccountability*, *supra* note 333, at 177.

code," which could potentially allow a remote intervention into the system.[406] Although Cisco corrected the flaw and ceased distributing the code that enabled the issue, Lynn remained concerned that Cisco had failed to do enough to encourage its customers to update its system and correct the error.[407] For this and other reasons, Lynn desired to give a presentation at Black Hat. When Cisco instructed him not to give the presentation, he quit his job, even though the presentation would not have provided enough detail to enable anyone to take advantage of the exploit without a great deal of effort.[408] Nevertheless, Cisco then sought a court order against Lynn, preventing him from presenting on the grounds that there was a risk that he would disclose Cisco's trade secrets to the public.

Although the case eventually settled with an agreement that Lynn would refrain from disseminating the information, it serves as a powerful example of the growing reliance on trade secrecy to impede the circulation of important public information. This case, according to David Levine, "meant that this information remained subject to laws designed to protect Cisco's interest, not the public's," running the risk that it would deter others from reverse engineering for fear of suffering the same fate.[409]

V
TOWARD CONTROLLED DISCLOSURE

As Frank Pasquale and others have explained, disclosure of source code is a deceptively simple solution to the problem of algorithmic transparency.[410] At best, it represents only a partial solution to the issue of accountability in AI because of the complexity and dynamism of machine-learning processes.[411] Many systems have also not been designed with oversight and

406 *Id.* at 178.

407 *Id.*; *see also* Jennifer Granick, *More Tales From 'Ciscogate'*, WIRED (Aug. 8, 2005), https://www.wired.com/2005/08/more-tales-from-ciscogate/ [https://perma.cc/HV6Q-ME29] (offering a first-hand account).

408 Levine, *Secrecy and Unaccountability*, *supra* note 333, at 178.

409 *Id.* at 180.

410 PASQUALE, *supra* note 347, at 142. See also Pasquale's work on qualified transparency in *Beyond Innovation and Competition: The Need for Qualified Transparency in Internet Intermediaries*, 104 NW. U. L. REV. 105, 162, 164 (2010) [hereinafter *Beyond Innovation*] (describing qualified transparency as an "excellent method" for creating a self-sustaining public).

411 Joshua A. Kroll et al., *Accountable Algorithms*, 165 U. PA. L. REV. 633, 638, 660 (2017). A code audit, sometimes referred to as "white box testing," can include examinations of, as one report describes, "specific system behavior—logs that record data access, calculations, decision trees, and errors," and, in some cases of automated systems, might include a review of the statistical models used

accountability in mind and, thus, can be opaque to the outside investigator.[412] Auditing, too, has significant limitations, depending on the technique.[413] Further, even if source code disclosure reveals some elements of a decision reached through automated processing, it cannot be fully evaluated without an accompanying investigation of the training data—why certain types of data were selected (or not), the choice of rules of operation, and the steps taken to validate the decision.[414] Transparency, then, does not mean interpretability.[415] And then there is the problem of the dynamic nature of algorithmic decision making, which often amplifies issues of opacity as well.[416]

All these critiques are certainly true in demonstrating that access to the source code is only one part of a larger issue of a lack of transparency in AI. However, at the same time, a legion of civil and criminal cases involving software have demonstrated that access to the source code is often an essential starting place in performing a full investigation or independent validation of an automated decision.[417]

As I have suggested, the seclusion of source code masks an underlying problem within intellectual property law that intellectual property reform alone cannot solve. The problem, essentially, is two-fold: one involves the dynamics of a closed, privatized system of governance, and the other involves the failure of intellectual property principles to incentivize harmonization and disclosure in cases of significant public interest. Both these issues have crystallized around source code secrecy as a major area of concern.

to rank, sort, and score inputs. See the excellent study by RIEKE ET AL., *supra* note 8, at 19.

412 Kroll et al., *supra* note 411, at 649–50. For a discussion of Kroll's article, see Pauline Kim, *Auditing Algorithms for Discrimination*, 166 U. PA. L. REV. ONLINE 189 (2017).

413 Kroll et al., *supra* note 411, at 650–52.

414 Brauneis & Goodman, *supra* note 338, at 130–31.

415 Brauneis & Goodman, *supra* note 338, at 131. For an excellent discussion of different types of transparency in automated decision making, see Cary Coglianese & David Lehr, *Transparency and Algorithmic Governance* 26 (U. Penn. Law Pub. Law & Legal Theory Research Paper Series, Research Paper No. 18-38), https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3293008 [https://perma.cc/2PQU-LRKR] (discussing fishbowl and reasoned transparency).

416 Brauneis & Goodman, *supra* note 338, at 131–32.

417 *See, e.g.*, Chessman, *supra* note 43, at 207 ("[A]ccess to source code is especially significant when evidence produced by a computer plays a prominent role in a defendant's trial . . . limiting source code access means . . . 'the defendant is effectively disabled from answering the one question every rational juror needs answered:' why does a computer think that you are guilty?" (footnotes omitted)).

Because of the complexity of the problem, we need to study a wide range of variables in reaching an individualized solution, interrogating the degree, depth, scope, timing, and audience of the disclosure.[418] Each of these elements will vary according to the type of issue presented, particularly whether it implicates state or privately sponsored deprivations of entitlements. At times, therefore, some limited disclosure—to experts, for example—might be more appropriate for investigative purposes.[419]

Source code is especially paradoxical, as I have argued, because its very nature is composed of both public and private property: many programming companies, as I have suggested, integrate open source code into their proprietary software.[420] Evidence suggests that over two-thirds of companies build proprietary software using open source code.[421] Other companies, as Chessman and others have pointed out, rely on code, algorithms, or software that draws from industry standards that are publicly available.[422] And yet, we have no way of knowing when a company asserts trade secret protection whether the underlying asset would satisfy the doctrinal definition.[423]

The problem, as I have suggested, is not just a problem of opacity—it may also implicate problems of privatization.[424] In such cases, building accountability does not simply mean ensuring greater transparency, it also encompasses, at times, some form of judicial review to ensure accountability as well.[425]

In addition, any menu of potential solutions must be situated within the background of the fluid nature of intellectual property protection for software, which I have argued has only served to heighten the attractiveness of trade secrecy protection for source code. At the same time, however, one must be pragmatic about the prospects for a solution. Patent protection for software is indeterminate and unlikely at best. And copyright law has largely bent over backwards to accommodate the secrecy of source code, essentially eviscerating its own system

[418] *Id.* For an excellent study of disclosure and its effects, see Bert I. Huang, *Shallow Signals*, 126 HARV. L. REV. 2227 (2013); see also Sandeen's excellent discussion of disclosure, *supra* note 282.
[419] PASQUALE, *supra* note 347, at 142.
[420] Chessman, *supra* note 43, at 210.
[421] *Id.* at n.224.
[422] *Id.* at 210.
[423] *Id.* at 209–10.
[424] As Ken Bamberger has observed in a related context, "even though the functions involved are traditionally those of a public actor, the management of those functions is private." Bamberger, *supra* note 15, at 726.
[425] *Id.* at 726–27.

of deposit requirements that serve the public interest. These scenarios make trade secrecy an especially attractive backup option, but they also impede a more systemic approach toward balancing the interests of property, privacy, and disclosure.

In this concluding section, I sketch out a brief architecture of what I would call a "controlled disclosure" regime—one that seeks to balance out the incentives at play in intellectual property, but one that also recognizes the pillars of discovery, disclosure, and open governance in order to address the growing issue of source code secrecy. This section explores a spectrum of solutions, from systemic to case-by-case solutions, which can be loosely clustered into "ex ante" solutions (which aim toward proactively incentivize disclosure of source code for limited public access) and those which might be construed as "ex post" solutions (which aim to particularize disclosure in a specific dispute). The idea here is to sketch out a wide range of tools for lawyers and litigators addressing these issues (recognizing, of course, that many of these are only superficial fixes to a deeper set of problems).

A. Strategies Toward Transparency

1. *Reforming Intellectual Property: Channeling and Election Doctrines*

The most systemic avenue of reform could involve addressing the current state of overlap between copyright, patent, and trade secret protection of software (and source code specifically). Here, the paradox of software secrecy is exacerbated by the longstanding judicial principle that the same aspects of software should not be protected by overlapping patent and copyright protections.[426] And yet when it comes to source code, or even broader aspects of software, more recently this overlap seems to be not only welcomed but also undertheorized.[427]

Most cases of overlap do not present a problem for intellectual property owners.[428] As Laura Heymann has explained, overlap is similar to a "belt-and-suspenders form of enforcement, allowing the intellectual property owner to resort to a second mode of protection should the first fail or expire."[429] In the case of software, as this Article has argued, the overlap

[426] Samuelson, *supra* note 14, at 1.
[427] *Id.* at 1, 3–4 (citing Oracle America, Inc. v. Google Inc., 750 F.3d 1339 (Fed. Cir. 2014) as evidence).
[428] *See* Heymann, *supra* note 134, at 240.
[429] *Id.* at 240.

(coupled with the shifting boundaries of protection) has led to a reliance on secrecy over disclosure, even in cases with strong public interest implications, largely because the law has facilitated it.

Scholars, including Mark McKenna and Christopher Sprigman, have recognized the role played by "channeling" doctrines, which operate to police the boundaries between various areas of intellectual property law, particularly with respect to subject matter.[430] The functionality doctrine in trademark law is a good example of this because it acts to ensure that aspects that are functional are "channeled" into patent, rather than trademark, law for protection.[431]

However, software—and the way that the law has governed it—lacks a comparable "channeling" influence, to the detriment of the public. Consequently, some have argued, particularly in the context of design patents, that an election doctrine may remedy the issue of overlap.[432] The same may be true here. As Christopher Buccafusco and others have described, the "election" requirement historically required that a creator choose a single form of protection for the work.[433] This view, they argue, stemmed from the court's perception that a work with multiple components may require that different regimes apply to these different parts.[434] Yet as Buccafusco points out, the absence of a doctrine for election "has increasingly meant that IP owners use different IP regimes to protect the *same aspects of the same works*, leading to overlapping protection."[435] This allows the IP owner to "leverage the advantages of all of these systems simul-

[430] Mark P. McKenna & Christopher Jon Sprigman, *What's In, and What's Out: How IP's Boundary Rules Shape Innovation*, 30 HARV. J.L. & TECH. 491, 542 (2017). The notion of channeling versus overlap has been addressed by scholars mostly in the context of design patents. *See* Mark P. McKenna, *An Alternate Approach to Channeling?*, 51 WM. & MARY L. REV. 873, 875–76 (2009); Heymann, *supra* note 134, at 240.

[431] McKenna, *An Alternate Approach to Channeling?*, *supra* note 430, at 876.

[432] *See* Christopher Buccafusco, Mark Lemley & Jonathan Masur, *Intelligent Design*, 68 DUKE L.J. 75, 81 (2018).

[433] *Id.* at 127 (citing Jason J. DuMont & Mark D. Janis, *U.S. Design Patent Law: A Historical Look at the Design Patent/Copyright Interface*, *in* THE COPYRIGHT/DESIGN INTERFACE: PAST, PRESENT & FUTURE 351 (Estelle Derclaye ed., Cambridge Univ. Press 2018)); Douglas R. Wolf, *The Doctrine of Elections*, 9 CARDOZO ARTS & ENT. L.J. 439 (1991) (noting that the doctrine of election has been "substantially abandoned").

[434] See Buccafusco et al.'s discussion of a 1974 case involving a watch design, *In re Yardley*, where the Court of Customs and Patent Appeals rejected the election doctrine. Buccafusco et al., *supra* note 432, at 128 (discussing In re Yardley, 493 F.2d 1389 (C.C.P.A. 1974)).

[435] *Id.* at 128.

taneously, rather than accepting the limitations of a given system as the price of obtaining its benefits."[436]

One solution, therefore, is to create a regime that essentially requires software owners to elect between doctrines, to force owners to choose at the outset a particular area of protection, or, relatedly, agree to relinquish one area of law if the owner selects one over the others.[437] One could imagine a channeling regime at the outset (when a creator seeks protection) or an election doctrine later on (if one chooses to litigate an infringement claim).

This argument has been made previously in the context of software, and it has intuitive appeal at first glance.[438] This framework for straightforward segregation would suggest that patents should protect functional implementation of concepts, copyright protects various modes of expression, and trade secrecy should be available for the protection of functional elements when patent protection is unavailable or undesirable.[439] Under an election-based theory, one's choice would be limited to individual features of a product, rather than the product as a whole, enabling software—which is a collection of various elements—to have different areas of protection, depending on the attribute that is being protected.[440]

Yet if we look closely, we see some difficulties with an election or channeling approach in the context of software. As many have pointed out, trade secrecy became a dominant form of protection not because of a pointed intellectual property strategy but because of the sheer mass of code that is out there, always changing, and because trade secret protection is so informal and easy to assert without challenge.[441] Moreover, it is also a powerful weapon in litigation, particularly compared to copyright, since claims do not require evidence of copying and can be narrowed further during discovery. Further, without a corresponding legislative fix that requires disclosure in the context of a deposit, a developer can still copyright code

---

[436] *Id.* at 128–29.

[437] Heymann, *supra* note 134, at 241.

[438] *See* Buccafusco et al., *supra* note 432, at 129 (citing Michael J. Kline, *Requiring an Election of Protection for Patentable/Copyrightable Computer Programs*, 6 COMP. L.J. 607 (1986)).

[439] Maier, *supra* note 145, at 151.

[440] See Buccafusco et al., *supra* note 432, at 132, making this argument in the context of design patents.

[441] *See, e.g.*, Maier, *supra* note 145, at 162 ("[I]t is clear that a computer program including logic, structure, and organization can qualify for trade secret protection as long as it is not generally known.").

without disclosing it.[442] And, given the lessons of history, a regime that requires full disclosure might actually have the opposite effect of incentivizing trade secrecy even further.

The same is effectively true for software patents, even if one obtains protection. Evidence shows that the Federal Circuit, even when it accepts software patents, has been loathe to require disclosure of source code as a precondition to patentability.[443] And there are other areas where patent law's requirements have been more lax than others. More than fifteen years ago, Dan Burk and Mark Lemley argued that recent patent law decisions had begun to demonstrate a striking willingness to excuse software inventions from the enablement and best mode requirements, limiting the goal of disclosure that is at the heart of the patent system.[444] In a variety of cases, the Federal Circuit held that software patentees need not disclose source or object code, flow charts, or other detailed descriptions of their programs.[445] The collective result of these cases, they argue, is an effective nullification of the disclosure requirement for software patents.[446] "[S]ince source code is normally kept secret," they explain, "software patentees generally disclose little or no detail about their programs to the public."[447]

[442] *See* Note, *Copyright Protection*, *supra* note 96, at 1740.

[443] *See* Ajeet P. Pai, Note, *The Low Written Description Bar for Software Inventions*, 94 VA. L. REV. 457, 479 (2008) (noting that there is a much lower bar for disclosures of software-related inventions compared to biotechnological inventions).

[444] *See* Dan L. Burk & Mark A. Lemley, *Is Patent Law Technology-Specific?*, 17 BERKELEY TECH. L.J. 1155, 1156 (2002).

[445] *Id.* at 1162–63 (discussing Northern Telecom, Inc. v. Datapoint Corp., 908 F.2d 931, 941 (Fed. Cir. 1990) and Fonar Corp. v. Gen. Elec. Co., 107 F.3d 1543, 1549 (Fed. Cir. 1997)). Indeed, Burk and Lemley point out that in multiple cases, the Federal Circuit has been so relaxed that it has permitted applicants to meet the requirements for written description and best mode, even when the specification fails to even use the terms "computer" or "software." *Id.* at 1164. Despite the relaxation of the requirements for disclosure, however, the authors are careful to point out that obviousness can be a rather tough bar for software patents to satisfy. *See id.* at 1167–68 (applying this analysis to Amazon's 'one-click' shopping feature). For more discussion of how obviousness operates in the context of software, see Jeanne C. Fromer, *The Layers of Obviousness in Patent Law*, 22 HARV. J.L. & TECH. 75, 95–98 (2008).

[446] Burk & Lemley, *supra* note 444, at 1164–65.

[447] *Id.* at 1165. In fact, one commentary from 1996 described source code listings as "primarily a relic of the early days of computer program patents when it was unclear what would suffice for sufficiency of disclosure." *See id.* at n.42 (citing MELVIN C. GARNER ET AL., *Advanced Claim Drafting and Amendment Writing Workshop for Electronics and Computer-Related Subject Matter*, *in* ADVANCED CLAIM AND AMENDMENT WRITING 1996, at 227, 275 (PLI Sixth Annual Patent Prosecution Workshop Course Book, 1996)).
Id. at 1165–66.

As a result, as a few leading experts in the field, Richard Stallman and Mitch Kapor, pointed out, even when software patenting was readily available, it did not affect the preexisting domains of trade secrecy.[448]

> By withholding the source code, companies keep secret not a particular technique, but the way that they have combined dozens of techniques to produce a design for a complete system. Patenting the whole design is impractical and ineffective. Even companies that have software patents still distribute programs in machine code only[,]

concluding that in no area do software patents reduce trade secrecy.[449]

2. *Reforming Copyright: Deposit and Demarcation Possibilities*

Even if a systemic approach is not available, what about reforming copyright law? As we know, the Copyright Office will register software without requiring the deposit of the source code and, generally speaking, copyright registration is only required when a person intends to file suit for infringement.[450] Deposit requirements, too, are not always enforced and remedies for noncompliance have been referred to as largely "toothless."[451] Nevertheless, this section discusses two possibilities: the first a system that reinvigorates disclosure through deposit, and the second a system that focuses on demarcating source code for discovery and other purposes.

First, given the indeterminate benefits of relying on fair use and reverse engineering in addressing source code secrecy, it makes sense to consider a simple legislative fix regarding source code protection in copyright law. Here, it may be worth revising copyright's formalities, like registration and deposit, in certain cases.[452]

As this Article has discussed, publication formalities abound in copyright law—with the notable exception of software.[453] Until 1976, federal protection under copyright could not attach until something was published, except in one context: source code, which is protectable without publication, comprehensive deposit or disclosure.[454]

448 Garfinkel et al., *supra* note 228, at 54.
449 *Id.* at 54.
450 *See* Note, *Copyright Protection*, *supra* note 96, at 1741 & n.120.
451 Gibson, *supra* note 20, at 208.
452 *See id.*
453 *See id.* at 205–06.
454 *Id.* at 206; *see infra* Part IC.

For various reasons, I would not favor a publication requirement for all forms of source code, even under copyright law. Complete transparency of code, particularly in cases where the source code addresses issues of vital public importance like electronic voting, requires some forms of seclusion and security to protect against hacking, gaming, or other forms of interference.[455] Moreover, a fully transparent society brings significant risks of invasions of privacy, voyeurism, and theft of intellectual property.[456] Even disclosures oriented to the public interest can become compromised by enabling other, less publicly-minded individuals to "game" or abuse the algorithm.[457]

At the same time that these concerns exist, it may make sense for us to revisit formalities nonetheless. For example, even if a uniform publication requirement seems unnecessarily overbroad and undesirable from a security perspective, there may be room to explore the possibility of a more pronounced deposit requirement with state officials or special masters in cases of strong public interest.[458] In such circumstances, it may be feasible to make the code available for inspection under certain circumstances warranting public interest.[459]

Indeed, the *sui generis* approach explored by Samuelson and others in their famous Manifesto argued that a registration and licensing system, coupled with an electronic repository for state-of-the-art software, would enable beneficial exchanges and facilitate low-cost transactions of software reuse.[460] A repository would facilitate greater public access, making more knowledge available to software engineers and benefiting the public as a result.[461] Others have also argued that a compulsory licensing regime might be appropriate for certain applications as well.[462]

---

[455] *Id.* at 206–07.

[456] PASQUALE, *supra* note 347, at 142.

[457] Sandvig et al., *supra* note 399, at 9 (noting that even Reddit, despite its culture of transparency, does not share all of its source code with the public).

[458] *See* Christopher Sprigman, *Reform(aliz)ing Copyright*, 57 STAN. L. REV. 485, 532–33 (2004).

[459] *See* Gibson, *supra* note 20, at 208–09. Indeed, *Ruckelshaus* expressly authorizes this sort of disclosure. *See* Ruckelshaus v. Monsanto Co., 467 U.S. 986, 1015–16 (1984) (allowing for government disclosure of trade secrets to eliminate research duplication and to streamline pesticide registration process).

[460] Samuelson, *supra* note 261, at 2425.

[461] *Id.* at 2429.

[462] *See* Peter S. Menell, *Tailoring Legal Protection for Computer Software*, 39 STAN. L. REV. 1329, 1371 (1987); Samuelson, *supra* note 261, at 2414–15; *see also* Anthony J. Mahajan, *Intellectual Property, Contracts, and Reverse Engineering After* ProCD*: A Proposed Compromise for Computer Software*, 67 FORDHAM L. REV.

Decades ago, the difficulty of finding the right mode of protection, it seems, actually motivated one agency in Japan, the Ministry of Industry and Trade, to propose a similar *sui generis* regime of protection that would last only fifteen years and required deposit of source code.[463] Most interestingly, it also proposed an arbitration system that empowered it to grant licenses to users when justified by the "public interest."[464] That proposal died as the result of negotiations with the U.S. Government, which expressed concern regarding its potentially lowered standard of protection.

Second, even aside from these options for limited disclosure, there are certainly middle pathways that can be explored in the marketplace and the courtroom.[465] It bears noting the curious parallel that emerges here between source code (which might not turn out to be as original as the developer might warrant) and the kind of concerns that animated the filtration/ abstraction tests that illuminated the early cases of nonliteral infringement. Just as the notion of nonliteral infringement implicates the risks of protecting more abstract work that comes from the public domain, literal infringement carries the same risk.

The idea, above, that is captured by the notion of filtration is that there is a spectrum of original and nonoriginal content in software. And for this reason, it may be possible to develop a demarcation system that offers some degree of openness to capture the complexity of code. Consider Creative Commons as an example, which in the copyright context enables a menu of options regarding openness for reuse.[466] Here, we could easily imagine public copyright demarcations that mark software according to: (1) full release of source code; (2) partial release of source code generally; (3) restricted release to certain parties.[467] Given the comparative popularity of the GPL model in open source projects, this may turn out to be an area of fruitful possibility.

3297, 3331–32 (1999) (suggesting a "compromise" between compulsory licensing and a complete ban on reverse engineering).

463 Oman, *supra* note 100, at 31.

464 *Id.*

465 For an excellent account of reinvigorating copyright's formalities, see Sprigman, *supra* note 458, at 554–64.

466 *See* CREATIVE COMMONS, https://creativecommons.org/about/ [https://perma.cc/T8NL-B2KS] (last visited Sept. 29, 2018) (introducing a network of copyright licenses allowing for greater customization).

467 This list of options modifies the very helpful framework set forth by Ince et al., *supra* note 388, at 487.

3. *Reforming Contract Law and Procurement*

Throughout this Article, I have mostly emphasized the intellectual property aspects of source code protection. There is, however, more to the story involving the role played by contract law in the early years of software's uncertain protectability. Since Congress did not amend the Copyright Act to include computer programs until 1980, and since patent protection emerged only after 1994 as a result of *In re Alappat*, trade secrecy became undergirded with a strong reliance on contract law (in the form of shrinkwrap licenses) for protection.[468] As a result, contract law has mostly been used to foreclose things like reverse engineering and imposing robust controls over subscribers that have been interpreted to foreclose some forms of auditing.[469] While this section begins by agreeing with many of the critiques of shrinkwrap license enforceability, I also wish to identify two potentially fruitful areas of challenge: the first involving a challenge of the Computer Fraud and Abuse Act to permit third-party auditing, and the second involving the potential for contractual reform with government entities.

Although there has been a healthy and robust debate regarding the enforceability of shrinkwrap licenses among scholars, courts mostly held the licenses unenforceable until the landmark Seventh Circuit case of *ProCD v. Zeidenberg*.[470] Al-

[468] *See* Mahajan, *supra* note 462, at 3297, 3310 & n.110.

[469] Lemley, *supra* note 267, at 1246–47.

[470] Mahajan, *supra* note 462, at 3310 (citing ProCD, Inc. v. Zeidenberg, 86 F.3d 1447 (7th Cir. 1996)). There is vast literature on the topic of enforceability. *See, e.g.*, David A. Einhorn, *Shrink-Wrap Licenses: The Debate Continues*, 38 IDEA 383, 401 (1998) (concluding that there is a circuit split regarding the enforceability of shrinkwrap contracts); Robert W. Gomulkjiewicz & Mary L. Williamson, *A Brief Defense of Mass Market Software License Agreements*, 22 RUTGERS COMP. & TECH. L.J. 335, 337 (1996) (arguing that courts and legislatures should validate the use of end user license (shrinkwrap) agreements); Mark A. Lemley, *Intellectual Property and Shrinkwrap Licenses*, 68 S. CAL. L. REV. 1239, 1253 n.53 (1995) (discussing the enforceability of shrinkwrap contracts); Mark A. Lemley, *Shrinkwraps in Cyberspace*, 35 JURIMETRICS J. 311, 317 (1995) (stating that "shrinkwrap licenses . . . do not fare well in the courts"); Apik Minassian, Comment, *The Death of Copyright: Enforceability of Shrinkwrap Licensing Agreements*, 45 UCLA L. REV. 569, 608 (1997) (arguing that the Seventh Circuit inappropriately applied the Copyright Act in *ProCD* to enforce the shrinkwrap agreement); Gary H. Moore & J. David Hadden, *On-line Software Distribution: New Life for 'Shrinkwrap' Licenses?*, 13 COMP. LAW. 1–10 (Apr. 1996) (arguing that online shrinkwrap licenses "stand[ ] a far greater chance of being enforced than [their] hard-copy cousin[s]"); Christian H. Nadan, *Software Licensing in the 21st Century: Are Software "Licenses" Really Sale, and How Will the Software Industry Respond?*, 31 AIPLA Q.J. 555, 640 (2004) (remarking that "only one significant case in the last five years has refused to enforce a shrinkwrap"); Maureen A. O'Rourke, *Drawing the Boundary Between Copyright and Contract: Copyright Preemption of Software License Terms*, 45 DUKE L.J. 479, 537 (1995) (arguing that shrinkwraps may be enforceable).

though I would definitely sympathize with the arguments regarding unenforceability, it is important to note the need for other avenues to protect researchers in their efforts to increase transparency through auditing. The world's leading computer science review community—the Institute of Electrical and Electronics Engineers (IEEE)—requires technically, managerially, and financially independent testing for any software that might cause "catastrophic consequences," defining that to include anything that causes a "[l]oss of human life, complete mission failure, loss of system security and safety, or extensive financial or social loss."[471]

Indeed, shifts in recent case law suggest growing areas of protection for independent auditing. Recently, the ACLU sued on behalf of four researchers who maintained that the Computer Fraud and Abuse Act—a national antihacking law—prevented them from scraping data from sites and from creating fake profiles to investigate algorithmic discrimination on the basis of race and gender.[472] The testers' concern was that the law permitted researchers to be vulnerable to criminal and contractual penalties because the research might involve violating one of the sites' Terms of Service.[473] Ironically, in real space, even though the use of crowdsourcing or human testers might be totally uncontroversial, the use of computer programs to replicate human behavior is often barred by contract.[474]

But the outcome of the *Sandvig* case was a strong statement in favor of protection for auditing techniques. There, the

[471] IEEE STANDARDS ASS'N, IEEE STD. 1012-2016: IEEE STANDARD FOR SYSTEM, SOFTWARE, AND HARDWARE VERIFICATION AND VALIDATION 196, 199 (2016); *see also* Nathaniel Adams, *What Does Software Engineering Have to Do with DNA?*, 42 CHAMPION 58, 65 (2018) (discussing importance of subjecting PG systems to software engineering best practices and independent reviews).

[472] In the case, two researchers attempted to run a sock puppet audit by creating a number of automated bots that would replicate the browsing habits of individuals of different races, and then visit a real estate web site and record the properties that they were shown and advertised. *See* Sandvig v. Sessions, 315 F. Supp. 3d 1, 8–10 (D.D.C. 2018); Complaint for Declaratory and Injunctive Relief at 23–24, Sandvig v. Lynch, No. 1:16-cv-01368 (D.D.C. June 29, 2016); *see also* Annie Lee, *Online Research and Competition Under the CFAA: The Revocation Paradigm of Interpreting Access and Authorization*, at 26–29 (draft on file with author); Sandvig et al., *supra* note 399, at 13.

[473] Complaint for Declaratory and Injunctive Relief at 24–25, *Sandvig*, No. 1:16-cv-01368.

[474] *See* Sandvig et al., *supra* note 399, at 15. Currently, federal courts disagree on the question of whether individuals who violate the Terms of Service restrictions can be prosecuted under the "access" provision of the CFAA, which provides for fines and punishment of anyone who "intentionally accesses a computer without authorization or exceeds authorized access, and thereby obtains . . . information from any protected computer" 18 U.S.C. § 1030(a)(2)(C) (2018).

court joined the Second, Fourth, and Ninth Circuits, which have stated that the CFAA prohibits "only unauthorized *access* to information" (e.g. hacking).[475] By narrowing the reach of the CFAA, the Court rejected a broader interpretation adopted by the First, Fifth, and Eleventh Circuits that the prohibited activities involved an unauthorized *use* of the information that went beyond authorization for specific purposes under the Terms of Service.[476]

Even outside of reforming the private, contractual nature of Terms of Service agreements and their interpretations, another potential area of success involves reforming contractual language with government parties.[477] In these contexts, contract law can serve as a tool for access, rather than the opposite. Previously, researchers reported that some cities, the City of San Francisco among them, rarely fought language in contracts with third-party vendors that recognized that the algorithms must be kept from the public.[478] Yet there is some evidence that this is changing and that more and more entities are looking to enhance openness through government contractual requirements that narrow, rather than expand, trade secrecy.[479]

In an important study performed by Robert Brauneis and Ellen Goodman, the authors note that "governments do not, and need not, uniformly accede to contractor wishes for nondisclosure and data ownership."[480] In Florida, for example, a pretrial risk assessment tool developed by the Arnold Foundation for use in the state court system was governed by contractual language that required the Foundation to specifically designate trade secret material or risk waiving the right to object to disclosure. By simply shifting the burden to the contractor to identify and mark its protected material, filtering out what is protected from what is public, the risk of overclaiming is reduced. It is also important to note that even though the Foundation would have preferred a broader scope of nondis-

[475] *Sandvig*, 315 F. Supp. 3d at 22–26 (emphasis added).

[476] *Id.* at 22.

[477] Joel Reidenberg discussed procurement as a potential avenue of reform in his landmark *Lex Informatica*, *supra* note 15, at 589.

[478] Abraham, *supra* note 1.

[479] See the work being done by Jason Schultz in advising government entities. E-mail from Jason Schultz to IPProfs (Nov. 28, 2018) (on file with author); ALGORITHMIC ACCOUNTABILITY POLICY TOOLKIT, AI NOW, 16–27 (2008), https://ainowinstitute.org/aap-toolkit.pdf [https:/perma.cc/4F62-N2YE].

[480] Brauneis & Goodman, *supra* note 338, at 164.

closure, it still readily agreed to a more transparent formulation.[481]

Indeed, as Brauneis and Goodman point out, when the government is doing the procuring, it is often in a more powerful position to ensure greater transparency. This means that governments can adopt a default position that presumes that all contractor-provided information is public in nature, or, alternatively, that the intellectual property produced under the contract is owned by the state. Indeed, in Illinois, their research revealed that at least one related contractor agreed to transfer ownership of all intellectual property rights under the contract to the state. In cases where a jurisdiction designs custom algorithms, the authors argue that it would be entirely appropriate to ask for ownership of the source code, or, at the very least, a nonexclusive license that authorizes the jurisdiction to authorize others; relatedly, a jurisdiction should also assert rights over any resulting reports that rely on particularized data.[482] In all cases, the authors urge jurisdictions to link their disclosure provisions to requests for full documentation, so that further investigation can take place if needed.

4. *Reforming Governance: Open Code Strategies*

The term "open code governance" was first used over twelve years ago by Danielle Citron in her sterling exploration of the topic to denote a world where source code used by government was publicly disclosed.[483] Yet, as I have argued in this Article, the issues that she raised with respect to closed code governance have only become further exacerbated in a world where algorithmic decision making replaces the norm of human judgment.[484] At the same time that she sounded the alarm on closed code in automated judgments, however, we have also seen a concomitant rise of commitment, at least during the Obama era, toward greater code transparency in government.[485]

In studying ways to reframe the source code paradox that is the central theme of this Article, we can turn to some of the core tenets of open government initiatives and see whether some examples might shed light on particular ways to en-

481 *Id.* at 165.
482 *Id.* at 166.
483 Citron, *supra* note 8, at 358.
484 *Id.* at 357–58.
485 *See Open Data Policy Guidelines*, SUNLIGHT FOUNDATION, https://sunlightfoundation.com/opendataguidelines/ [https://perma.cc/8NT2-QBSP] (last visited Sept. 29, 2018).

courage greater transparency.[486] A recently proposed law in Washington State called for public agencies to compile an "algorithmic accountability" report, requiring the system (and its data) to be available for independent verification, testing, and research to understand the potential for bias, inaccuracy, or disparate impact.[487] This is a perfect example of how law can address the problem of opacity to enable better transparency.

In the criminal justice context, Erin Murphy has proposed a system that would empower a centralized national oversight board to review and ensure defendants' access to private or proprietary data regarding certain forensic techniques.[488] The City Council law—and the accompanying hearing—that opened this Article is just one example of a growing and larger trend toward more openness in government through requiring source code disclosure and enabling black box testing (which allowed for mechanisms to test inputs and generate results (outputs)).[489]

One part of "technological due process"[490] (to use Citron's language), for example, might involve the creation of interactive models that allow citizens to see how certain decisions might change according to the input of a changing continuum of vari-

486 For more on transparency, see generally OPEN GOVERNMENT: COLLABORATION, TRANSPARENCY, AND PARTICIPATION IN PRACTICE (Daniel Lathrop & Laurel Ruma eds., 2010) (discussing online tools for government transparency and participation); Mark Fenster, *The Transparency Fix: Advocating Legal Rights and Their Alternatives in the Pursuit of a Visible State*, 73 U. PITT. L. REV. 443, 480 (2012) (analyzing transparency advocacy campaigns); *Free & Open Source Software in Government with Code.mil*, DIGITALGOV (June 5, 2018), https://digital.gov/event/2018/06/05/free-open-source-software-in-government-with-codemil/ [https://perma.cc/YF4J-6GV6] (discussing Code.mil, an effort to catalog open source efforts within the Department of Defense); *Open Government Initiative*, THE WHITE HOUSE: PRESIDENT BARACK OBAMA, https://obamawhitehouse.archives.gov/open [https://perma.cc/76ZP-YMW2] (last visited Sept. 29, 2018) (discussing President Obama's Open Data Initiatives).

487 *See* DJ Pangburn, *Washington Could Be the First State to Rein in Automated Decision-Making*, FASTCOMPANY (Feb. 8, 2019), https://www.fastcompany.com/90302465/washington-introduces-landmark-algorithmic-accountability-laws [https://perma.cc/NEK7-5NV8].

488 *See* Erin Murphy, *The New Forensics: Criminal Justice, False Certainty, and the Second Generation of Scientific Evidence*, 95 CALIF. L. REV. 721, 783–84 (2007).

489 *See* Testimony of Helen Nissenbaum, Julia Powles & Thomas Ristenpart, Hearing on Automated Decision Systems Used by Agencies, *supra* note 3, at 1 [hereinafter Nissenbaum et al.] (follow "Hearing Testimony 10/16/17" hyperlink, page 41 of pdf); *see also* Brauneis & Goodman, *supra* note 338, at 164–75 (discussing other ways in which governments can "promote transparency in their use of predictive algorithms").

490 *See* Danielle Keats Citron, *Technological Due Process*, 85 WASH. U. L. REV. 1249, 1260–67 (2008).

ables.[491] Or it might involve the creation of audit trails that enable individuals to see notice of the basis of automated government decision making, particularly where public benefits are concerned.[492] That is why the New York City Council law was so significant, because it aimed for a level of accountability that had not yet been demonstrated at the hands of local government. As a group of professors explained:

> A Bill like this has the potential to address several stark gaps in our regulatory landscape. When data is fed into a computer system and used to allocate public services, penalties, or policing, people deserve to know that the system is functioning in accordance with the City's aims and values. That it is not arbitrary, unfair, or incorrect. That it does not amplify inequality. This means being able to find out what data is used, how it is processed, and what else is taken into consideration in decision-making, both in general and in individual cases. There should be opportunities to test and contest the input, processing, and output.[493]

Of course, that does not mean that the Bill solved every issue of government opacity. For example, it failed to offer any degree of transparency regarding the data that was being used by an automated system, among other areas of oversight.[494] Nor did its commitment to transparency take precedence over proprietary claims in every instance.[495]

Nevertheless, there are powerful reasons for a commitment to open code, particularly in areas of governance, but even in private industry. Open source advocates argue that greater exposure to diverse minds will only improve the code, benefiting innovation more broadly.[496] And the market often favors open source projects as well, like the Apache web server, the Linux operating system, or the GNU Compiler Collection (GCC), which contains a variety of widely-used compilers for use with various programming languages.[497] Even Microsoft has a

---

491 *See* Brauneis & Goodman, *supra* note 338, at 174–75. In Europe, the GDPR has provided individuals with a "right to explanation." *See* Commission Regulation 2016/679, 2016 O.J. (L 119) Recital 71, Art. 13, Art. 15, Art. 22; Bryan Ware, *Is the 'Right to Explanation' in Europe's GDPR a Game-Changer for Security Analytics?*, CSO (Jan. 29, 2018), https://www.csoonline.com/article/3251727/is-the-gdpr-s-right-to-explanation-a-game-changer-for-security-analytics.html [https://perma.cc/7HE2-J6JV].

492 *See* Danielle Keats Citron & Frank Pasquale, *The Scored Society: Due Process for Automated Predictions*, 89 WASH. L. REV. 1, 28 (2014).

493 Nissenbaum et al., *supra* note 489, at 1–2.

494 See *id.* for an excellent discussion.

495 *Id.*

496 *See Source Code Definition*, *supra* note 45.

497 *Id.*

shared source initiative, which enables a select group of researchers, universities, and government actors to view selected portions of the Microsoft code (albeit under restricted conditions).[498]

Part of these initiatives, understandably, are motivated by the desire for better software security.[499] But part of it might also serve as an example to other entities about ways to share code responsibly with known parties. Frank Pasquale has also proposed making algorithms available to expert third parties who would essentially hold them in escrow, thus allowing them to be studied but not made public.[500]

Indeed, during the Obama era, the government sought to develop a number of open government initiatives to support ideas of transparency, participation, and collaboration.[501] Back in 2009, the Department of Defense (DOD) issued a groundbreaking memorandum that articulated a clear commitment to Open Source Software, requiring that executive agencies conduct market research and, in justified cases, prefer open source software over other choices, due to cost and other considerations.[502] It touted open source's added reliability and security, due in no small part to its "continuous and broad peer-review."[503] The memo also predicted that the ease of modifying open source would enable the DOD "to respond more rapidly to changing situations."[504]

Seven years later, in August 2016, the government released its Federal Source Code Policy, which required that all new custom source code be shared with other agencies for reuse, and that at least 20% of all new government custom

---

498 *See id.* (discussing Microsoft's Shared Source Initiative). *But see* Anne-Kathrin Kuehnel, *Microsoft, Open Source and the Software Ecosystem: Of Predators and Prey—The Leopard Can Change Its Spots*, 17 INFO. & COMM. TECH. L. 107, 107 (2008) (questioning whether Shared Source is truly a step toward an embrace of Open Source philosophy).

499 *See Source Code Definition, supra* note 45, at 5.

500 *See* Sandvig et al., *supra* note 399, at 9 (citing Pasquale, *Beyond Innovation, supra* note 410).

501 *See* Fenster, *supra* note 486, at 483; Norm Eisen & Ben Noveck, *Why an Open Government Matters*, THE WHITE HOUSE: PRESIDENT BARACK OBAMA (Dec. 9, 2009, 3:16 PM), https://obamawhitehouse.archives.gov/blog/2009/12/09/why-open-government-matters [https://perma.cc/PAW8-TVLW].

502 *See* Memorandum from Dep't of Def., Clarifying Guidance Regarding Open Source Software (OSS) 4 (Oct. 16, 2009), http://dodcio.defense.gov/Portals/0/Documents/FOSS/2009OSS.pdf [https://perma.cc/4LPT-PEWC] (noting that OSS met the definition of "commercial computer software" in almost all cases and should be afforded a statutory preference in market research).

503 *Id.*

504 *Id.*

code be released to the public as open source software.[505] It also created code.gov as a way to encourage greater citizen participation and to release its open source projects to the public.[506] Although the comprehensive nature of the memo took some by surprise, open source projects had been percolating for years before at the FDA, DOD, and CFPB.[507]

Of course, the core objection to open code governance has to do with the political tides, which often turn in either direction. Consider the fate of open code governance in our current Federal Administration. Although the lead government official on the project, Alvand Salehi, argued that this was not a partisan issue, and observed "Code.gov is here to stay," there are few signs suggesting that the current administration has prioritized the issue, even though the web site still exists.[508]

---

505 *See* Nicole C. Baratta, *Sharing America's Code*, OPENSOURCE.COM (May 18, 2017), https://opensource.com/article/17/5/sharing-americas-code [https://perma.cc/RB7N-PLEP].

506 *Id.*

507 *See* Memorandum from Tony Scott & Anne E. Rung to the Heads of Dep'ts & Agencies, *Federal Source Code Policy: Achieving Efficiency, Transparency, and Innovation Through Reusable and Open Source Software* § 2, https://sourcecode.cio.gov/ [https://perma.cc/N4CL-XWMD] (last visited Sept. 29, 2018); *see also Petitions*, GITHUB, https://github.com/WhiteHouse/petitions [https://perma.cc/H4TM-TDS8] (last visited Sept. 29, 2018) (discussing Obama administration's decision to release source code for application that allows individuals to directly petition governments); *Petitions Under the Obama Administration*, THE WHITE HOUSE: PRESIDENT BARACK OBAMA, https://petitions.obamawhitehouse.archives.gov/[https://perma.cc/D4QK-QTPR] (last visited Sept. 29, 2018) (implementing APIs which enable users to gather petition signatures on third-party platforms)). Even the FDA built *OpenFDA*, U.S. FOOD & DRUG ADMIN., http://open.fda.gov [https://perma.cc/56MH-WNYX] (last visited Sept. 29, 2018), which was an API that enabled individuals to inquire about adverse drug reactions. The Consumer Financial Protection Bureau also used open source software. *See* Matthew Burton, *The CFPB's Source Code Policy: Open and Shared*, CONSUMER FINANCIAL PROTECTION BUREAU (Apr. 6, 2012), https://www.consumerfinance.gov/about-us/blog/the-cfpbs-source-code-policy-open-and-shared/ [https://perma.cc/LL5E-2BAE].

508 *See* Tom Cochran, *Farewell to Obama, Our First Digital President*, RECODE (Dec. 1, 2016, 8:00 AM), https://www.recode.net/2016/12/1/13765002/president-obama-digital-trump-administration-open-source [https://perma.cc/2M3X-5YMN] ("It is imperative that our government work with best-of-breed services and technologies to move our nation forward, and the introduction of open source models has allowed our government to do just that."); Alex Handy, *As Trump Moves in, Code.gov Appears to Leave*, SOFTWARE DEV. TIMES (Jan. 20, 2017), http://sdtimes.com/code-gov/trump-moves-code-gov-appears-leave [https://perma.cc/ERD9-GGCR] (noting that code.gov was down for a short period, then returned to full functionality); Clare Malone, *How Trump's White House Could Mess with Government Data*, FIVETHIRTYEIGHT (Dec. 15, 2016, 6:29 AM), https://fivethirtyeight.com/features/how-trumps-white-house-could-mess-with-government-data/ [https://perma.cc/32XG-J9A3] (discussing the possibility that the practices of the Trump administration will "erode the quality of government data collection and systems").

Nevertheless, aside from potential inertia at the federal level, it appears that many municipalities are well underway in opening up their code. One example that is particularly instructive involves efforts by municipalities to adopt information technologies and policies to make their data sets available to the public.[509] For example, the New York City Mayor's Office of Data Analytics (MODA) uses transparent, open source code for its data analytics and makes many of its projects public.[510] In October 2016, Boston launched boston.gov, releasing its source code to the public and promising the public that anything it builds going forward will be "open by default."[511] New York and its Metropolitan Transit Authority even set up a contest for software developers who develop apps based on government data sets.[512] San Francisco enacted the first open-data ordinance requiring city departments to make their data sets open to the public.[513]

Aside from private and public initiatives toward shared source, policymakers might also explore a more robust engagement by government into creating incentives for more open code initiatives. Ken Bamberger's excellent work on risk management technologies proposes the idea of regulators who might issue forms of "approval regulation," in which he describes a process by which technology providers would offer full transparency regarding their particular technologies in exchange for some form of legal safe harbor.[514] Or we could imagine a world by which government funding decisions would be explicitly tied to more transparent forms of governance or data-sharing with third parties to ensure greater accountability.[515]

[509] Fenster, *supra* note 486, at 484–85; *see also* Jennifer Shkabatur, *Cities @ Crossroads: Digital Technology and Local Democracy in America*, 76 BROOK. L. REV. 1413, 1443 (2011) (addressing efforts by municipalities to provide digital services).

[510] *See* Testimony of Don Sutherland, DEP'T OF INFO. TECH. & TELECOMM., at 2 (Oct. 16, 2017) (on file with author).

[511] *See* Ben Miller, *What's New in Civic Tech: Uncertainty in the Age of Trump, Open Source Projects Abound*, GOV'T TECH. (Nov. 10, 2016), http://www.govtech.com/civic/Whats-New-in-Civic-Tech-Uncertainty-in-the-Age-of-Trump-.html [https://perma.cc/6VAN-HQRB]. For a great discussion of various open-code projects in governance, see OPEN GOVERNMENT, *supra* note 486.

[512] Fenster, *supra* note 486, at 484.

[513] *Id.* at 485.

[514] *See* Bamberger, *supra* note 15, at 736.

[515] *See* Nissenbaum et al., *supra* note 489.

B. Strategies Toward Disclosure

1. *Reforming Trade Secrecy: Identification and Filtration*

As this Article has suggested, part of the issue that inspires the paradox of source code secrecy stems from a fundamental problem regarding an overbroad delegation of authority to the trade secret owner. The identification of a trade secret is an incredibly subjective determination, and the plaintiff essentially enjoys total deference in deciding what to include and how to describe the matter at issue.[516] Indeed, even the factors that are normally relied upon to determine whether a trade secret exists (the extent to which the information is known outside and inside the business; the extent of measures taken to protect the secrecy of the information; its value and its cost of development; and the ease with which it could be acquired or duplicated by others) have little to do with the underlying substance of what is protected.[517]

As a result, courts display a systemic tendency to conflate the question of whether a plaintiff has identified an alleged secret with the question of whether the information is *actually* a trade secret.[518] Without a precise identification of the source code elements, a defendant is essentially prevented from comparing the claims against information in the public domain, thereby hampering their defense.[519]

In a fascinating, comprehensive study, two software lawyers, Tait Graves and Brian Range, explained that it is "common for a trade secret plaintiff to alter its list of trade secret claims as the case proceeds—sometimes dramatically, by replacing entire categories of information or technology, or by recombining slippery, multi-element 'combination trade secret' claims into new subsets."[520] For example, the plaintiff might claim an "entire process"—consisting of its entire source code,

516 *See* Graves & Range, *supra* note 393, at 73.

517 *See, e.g.*, GlobeRanger Corp. v. Software AG, 836 F.3d 477, 492 (5th Cir. 2016) (listing factors); RESTATEMENT (FIRST) OF TORTS § 757 (AM. LAW INST. 1939) (outlining the factors). To determine whether a trade secret exists, the Restatement dictates examination of six factors: "(1) the extent to which the information is known outside of his business; (2) the extent to which it is known by employees and others involved in his business; (3) the extent of measures taken by him to guard the secrecy of the information; (4) the value of the information to him and to his competitors; (5) the amount of effort or money expended by him in developing the information; (6) the ease or difficulty with which the information could be properly acquired or duplicated by others." RESTATEMENT (FIRST) OF TORTS § 757 cmt. b (AM. LAW INST. 1939).

518 *See* Graves & Range, *supra* note 393, at 71–72.

519 *Id.* at 68–69.

520 *Id.* at 68.

or its entire chip design—or it might revise its claim in different mixes of subsets, what Graves and Range refer to as "gerrymander[ing] a claim so that the defense cannot focus its research efforts on defeating the final version."[521]

In one representative case from the Fifth Circuit, the court concluded that the plaintiff had produced enough evidence for a jury to conclude that "at least some portion of its . . . [source code] constituted a trade secret."[522] The court simply reached its conclusion based on assertions regarding the uniqueness of the technology and its reliance on restrictions on the source code's circulation.[523] At the end of the day, the court's reasoning risks becoming somewhat circular in nature: something is secret because it is said to be secret, not because the information, in actuality, *is* secret or because its secrecy is proven with particularity.[524]

Certainly, more nuance or more willingness on the part of courts to examine the material would be very valuable for two reasons: first, as a substantive check on the nature of what is claimed to be protected, and second, as a signaling function to suggest that courts may be less deferential to future claimants.[525] Without a precise identification of the elements of software code or hardware architecture, a defendant is unable to compare the claims against information that is already in the public domain and therefore is unable to mount an effective challenge.[526] The problem is made even worse by the fact that courts rarely quote descriptions of trade secrets, and therefore many published opinions do not serve as guides for others to follow.[527]

---

521 *Id.* at 77. They further explain: "If we take, for example, seven software algorithms and assume that five are in the public domain, the plaintiff might alter the claim several times to create subsets of the seven where at least one of the included algorithms is secret, in order to claim the non-secret algorithms as secret as well." *Id.*

522 *GlobeRanger*, 836 F.3d at 492.

523 *Id.* (listing the six factors set forth in Restatement (First) of Torts § 757 (AM. LAW INST. 1939)). Note that the six-factor test is arguably obsolete now, given the increased reliance on UTSA and DTSA factors in jury instructions. *See* Correspondence from Tait Graves to author (Jan. 27, 2019) (on file with author).

524 *GlobeRanger*, 836 F.3d at 492.

525 *See generally* Bertelsen v. Allstate Ins. Co., 796 N.W.2d 685, 703–05 (S.D. 2011) (holding that "claims manuals, training materials, and salary administration materials" constituted protected trade secrets); In re Bass, 113 S.W.3d 735, 740 (Tex. 2003) (holding that geological seismic data involving the land was a protected trade secret).

526 *See* Graves & Range, *supra* note 393, at 68–69.

527 *Id.*

I would argue for a more nuanced approach to literal forms of infringement regarding source code, and one that might interpret questions of source code protection through the lens of the filtration tests outlined in the previous generation of software case law.[528] Expert testimony, for example, is used under *Altai*.[529] *Gates*, for example, emphasized the importance of filtering out all unoriginal elements of a program, and there is no reason not to subject source code to a more aggressive mode of filtration as well.[530] Further, in the non-software context, there is mounting case law that requires parties to describe, define, and identify, with increased particularity, the trade secrets in question rather than offer a blanket assertion of confidentiality, even before the expert discovery process has commenced.[531] To some extent, some of that nuance is already starting to occur in some software infringement cases, though not yet in the criminal context.

One powerful solution to address the issue of trade secret identification could be to adopt California's version of the Uniform Trade Secrets Act, which requires trade secret plaintiffs to provide a reasonably particular identification of alleged secrets prior to pursuing discovery and provides remedies for bad faith trade secret claims.[532] Courts in several states—Delaware, Illi-

---

528 *See* Gen. Universal Sys. Inc. v. Lee, 379 F.3d 131, 142–43 (5th Cir. 2004) ("[T]he court filters out unprotectable expression by examining the structural components at each level of abstraction to determine whether they can be protected by copyright."); Gates Rubber Co. v. Bando Chem. Indus., 9 F.3d 823, 836 (10th Cir. 1993) (noting that a court "must filter out those elements of the program that are not protected by copyright").

529 *See* Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 712–13 (2d Cir. 1992); Samuelson, *supra* note 147, at 1770–71 (noting that filtration narrows the scope of copyright protection by removing "public domain elements of programs, such as commonplace programming techniques, ideas, and know-how").

530 *See Gates Rubber Co.*, 9 F.3d at 837–38 (citing Comprehensive Techs. Int'l, Inc. v. Software Artisans, Inc., 3 F.3d 730, 736 (4th Cir. 1993); *Comput. Assoc. Int'l*, 982 F.2d at 710; Brown Bag Software v. Symantec Corp., 960 F.2d 1465, 1474–75 (9th Cir. 1992); E.F. Johnson Co. v Uniden Corp. of Am., 623 F. Supp. 1485, 1499 (D. Minn. 1985)).

531 *See, e.g.*, Synygy, Inc. v. ZS Assocs., Inc., No. 07-3536, 2015 WL 899408, at *6–9 (E.D. Pa. Mar. 3, 2015) (requiring further definition of the scope of a trade secret during discovery);*see also* Michael P. Broadhurst & Ann E. Querns, *Define Trade Secrets Before and During Litigation*, BLANKROME (May 12, 2015), https://www.blankrome.com/index.cfm?contentID=37&itemID=3582 [https://perma.cc/DU5W-LZ2N] ("A series of decisions in *Synygy v. ZS Associates*, No. 07-3536 (E.D. Pa. March 3, 2015), highlight the critical importance of defining an enterprise's trade secret information . . . .").

532 *See* Graves & Range, *supra* note 393, at 71, 76, 83; CAL. CIV. CODE § 3426.1(d) (2018) (defining trade secret under California Uniform Trade Secrets Act). Massachusetts has also adopted a similar statute. *See* MASS. GEN. LAWS ch. 93, § 42D(b) (2018) ("In an action . . . alleging trade secrets misappropriation a

nois, Massachusetts, Minnesota, and possibly Florida, among others—have adopted similar requirements.[533] But so far, California is the only state to codify its rule, enacted in part because of its concern over discovery abuses engaged in by trade secret plaintiffs.[534] In a 2005 case, a California appellate court observed:

> The letter and spirit of section 2019.210 require the plaintiff, subject to an appropriate protective order, to identify or designate the trade secrets at issue with "sufficient particularity" to limit the permissible scope of discovery by distinguishing the trade secrets "from matters of general knowledge in the trade or special knowledge of those persons . . . skilled in the trade." . . . Where, as here, the alleged trade secrets at issue consist of incremental variations on, or advances in the state of the art in a highly specialized technical field, a more exacting level of particularity may be required to distinguish the alleged trade secrets from matters already known to persons skilled in that field.[535]

Even when the discovery and trial process unfolds, Graves and Range evince a strong set of recommendations that force the plaintiff to be specific in identifying its alleged secrets, including directing courts to be wary of high-level, general lists of trade secrets.[536]

In one influential California case, *Altavion v. Konica Minolta Systems, Laboratory Inc.*, the court spent a fairly long time exploring the adequacy of the trade secret identification.[537] There, even as it offered a broad and inclusive take on trade secrecy, it also drew up three tiers of "specificity and secrecy," ranging from the most secret (source code) to the least secret (the idea of the use of barcodes to enable self-authentication of documents).[538] At the middle tier were the design concepts used for the company's digital stamping technology, which could be ascertained by an end user but were still protectable as trade secrets.[539] What is instructive about that case is the court's willingness to delve into the substance of what consti-

party must state with reasonable particularity the circumstances thereof, including the nature of the trade secrets and the basis for their protection.").

533 *See* Graves & Range, *supra* note 393, at 82 (collecting examples).

534 *See id.* at 83 (noting the legislative discussion of abuses).

535 *See id.* at 84 (quoting Advanced Modular Sputtering, Inc. v. Superior Court, 132 Cal. App. 4th 826, 835–36 (2005)).

536 *See* Graves & Range, *supra* note 393, at 91–96.

537 Altavion, Inc. v. Konica Minolta Sys. Lab. Inc., 226 Cal. App. 4th 26, 43–46 (2014).

538 *Id.* at 56.

539 *Id.* at 48–49.

tuted the protectable trade secret as opposed to simply deferring to the owner. One could imagine a situation where an expert might postulate particular tests for filtering the public domain, open source content from its protectable, secret matter.

Since trade secrecy can attach to a number of different aspects of code—object code, algorithms, information or formulas detailed in source code, software architecture, and data structure, among other categories, Graves and Range recommend identifying each category specifically, and sequestering it from auto-generated code, open source material, or basic code that is mandated by the type of program, because all of that information is already nonsecret in nature.[540] In addition, the lawyers recommend that the plaintiff literally specify the exact lines of code claimed to be secret by identifying the allegedly misappropriated lines by number or highlighting.[541]

Other strategies might involve requiring the plaintiff to reference how much of the source code already remains in the public domain, in addition to considering the conventional factors to assess trade secret protection.[542] Indeed, on the question of source code discovery and the public domain, courts can exercise greater scrutiny.[543] In one civil case, a court required a plaintiff to explicitly identify the trade secret components of the source code, reasoning that merely providing the defendants with a "reference library" to establish what portions of the code were in the public domain impermissibly shifted the burden to the defendants. The court quoted from an earlier case that made the argument that:

> [A] plaintiff "ha[s] to be able to identify with specificity what information [it] consider[s] to have been a trade secret[.] . . . If the plaintiff can't do that now, it can't proceed on that theory, because the defendants have a right during discovery to test whatever the plaintiff's theory is. . . . Plaintiff is the only one who can know what it believes its trade secrets are. . . . And it

540 *See* Graves & Range, *supra* note 393, at 93–95.

541 *See id.* at 94–95.

542 Parsons v. Pa. Higher Educ. Assistance Agency, 910 A.2d 177, 184–85 (Pa. Commw. Ct. 2006).

543 One court has held that it is plainly insufficient for a plaintiff to establish source code protection by identifying only those aspects of its source code that were *not* trade secrets because they were in the public domain, covered by third party licenses, or unprotected. MSCI Inc. v. Jacob, 945 N.Y.S.2d 863, 864–66 (N.Y. Sup. Ct. 2012).

> is unfair to . . . the defendants to conduct discovery without knowing what the assertions are."[544]

By requiring greater identification and particularity, judges can empower more transparency in litigation, effectively increasing access to source code that already lies within the public domain.

2. *Reforming Discovery: Toward Controlled Disclosure*

A final, modest set of solutions focuses on invoking the familiar themes of discovery and disclosure, enabling greater procedural due process through transparency while recognizing the very liberal use of protective orders in trade secret cases.[545] It is well settled in IP cases that a trade secret holder can either establish a privilege to ensure seclusion or obtain a protective order to avoid disclosure to the public.[546] The task before us is to ensure that this principle translates to issues that implicate the public interest in transparency, particularly where automated decision making is concerned.

Typically, the burden rests with the party resisting discovery to show that the requested information is a trade secret.[547] After the owner shows that its disclosure would be harmful, the burden then shifts to the opposing party to show that the trade secret is relevant and necessary to prepare the case for trial.[548] The idea is to ensure that each party can effectively litigate its case, compelling discovery in situations where judicial resolution would be impossible but for the substance of the trade secret.[549]

Courts generally prefer not to deny discovery merely because of the risk that the trade secret will be disclosed, but instead will try to consider the interests of both parties and the

[544] *Id.* (quoting Sit-Up Ltd. v. IAC/InterActiveCorp., No. 05 Civ. 9292 (DLC), 2008 WL 463884, at *6 (S.D.N.Y. Feb. 20, 2008)).

[545] *See* Stadish v. Superior Court, 84 Cal. Rptr. 2d 350, 359 (Cal. Ct. App. 1999) (noting utility of protective orders in trade secret case).

[546] *See* Pincheira v. Allstate Ins. Co., 190 P.3d 322, 330 (N.M. 2008).

[547] *See* Sea Coast Fire, Inc. v. Triangle Fire, Inc., 170 So. 3d 804, 808 (Fla. Dist. Ct. App. 2014).

[548] *Id.* at 809.

[549] *See id.*; MSCI Inc. v. Jacob, 945 N.Y.S.2d 863, 864 (N.Y. Sup. Ct. 2012). In such cases, the party requesting the information has to show "how the lack of the information will impair the presentation of the case on the merits to the point that an unjust result is a real, rather than a merely possible, threat." In re Goodyear Tire & Rubber Co., 392 S.W.3d 687, 696 (Tex. App. 2010) (quoting In re Bridgestone/Firestone, Inc., 106 S.W.3d 730, 732–33 (Tex. 2003)); *see also* Laffitte v. Bridgestone Corp., 674 S.E.2d 154, 163–64 (S.C. 2009) (finding that plaintiff's experts did not establish the specific need for disclosure of formula of rubber tire composition).

interests of justice.[550] As Graves has observed, in most civil cases involving trade secrets, protective orders are effectively mandated "so the concept that [a] claimed [trade secret] is discoverable is already, implicitly, decided."[551] At the same time, however, litigation around discovery matters can be costly and maddening at the same time. As former Judge Grewal has observed:

> In a typical patent infringement case involving computer software, few tasks excite a defendant less than a requirement that it produce source code. Engineers and management howl at the notion of providing strangers, and especially a fierce competitor, access to the crown jewels. Counsel struggle to understand even exactly what code exists and how it can be made available for reasonable inspection. All sorts of questions are immediately posed. Exactly who representing the plaintiff gets access—and does this list include patent prosecution counsel, undisclosed experts, and so-called "competitive decision makers"? Must requirements and specification documents that explain the functionality implemented by the [test] code be included? What compilation, debugging and analysis tools are required? What about the test database and user manuals? Make files? Build files? . . . Put simply, source code production is disruptive, expensive, and fraught with monumental opportunities to screw up."[552]

While Grewal added a note of humor to the monumental task of source code discovery, his observations offer two key insights. First, while source code production can be maddening, time consuming, and costly, it is by now relatively common in software cases.[553] Second, given that source code production is not an uncommon occurrence, litigators have ready-made tools at their disposal to address the merit of software-related disputes while ensuring that the source code remains

550 *See* Bleacher v. Bristol-Myers Co., 163 A.2d. 526, 528–29 (Del. Super. Ct. 1960) (collecting cases).

551 *See* Correspondence from Graves to author (Jan. 27, 2019) (on file with author).

552 Andrew Schulman, *Source Code ch.09: Discovery*, SOFTWARE LITIGATION CONSULTING, http://www.softwarelitigationconsulting.com/source-code-book/source-code-ch-09-discovery/ [https://perma.cc/M9U2-72D2] (last visited Oct. 4, 2018) [hereinafter Source Code & Software Patents] (quoting Apple Inc. v. Samsung Elecs. Co., No. C 11-1846 LHK (PSG), 2012 WL 1595784, at *1 (N.D. Cal. May 4, 2012)).

553 The Northern District of California has developed a model protective order source code, available at *Model Protective Orders*, UNITED STATES DISTRICT COURT: NORTHERN DISTRICT OF CALIFORNIA, https://www.cand.uscourts.gov/model-protective-orders [https://perma.cc/LW2K-B648] (last visited Apr. 26, 2019).

protected and yet disclosed in a litigation dispute.[554] Parties are by now familiar with drafting protective orders and other litigation-related tools to protect the seclusion of source code. There is no need to reinvent the wheel; it has already been turning for decades.

It is well-settled that courts can typically easily safeguard trade secrets during litigation; preliminary relief, like preliminary injunctions and TROs to prevent disclosure, is often granted.[555] Given the above, the lack of disclosure in the criminal context is particularly striking. If the matter at issue were about patent infringement, for example, where money and the marketplace were at stake, a court would routinely allow for further investigation and order the source code to be turned over to opposing counsel.[556]

It is important to note that a majority of the most stringent limitations of trade secret disclosure comes from a primary concern—competition—that is not always at issue in the contexts I have discussed. As one court explains, "[t]he main concern of parties seeking to impose AEO [Attorney Eyes Only] restrictions is fear that dissemination of sensitive information, particularly to decision-makers of its competitors, would threaten serious competitive harm."[557] If this is true, then the investigative (rather than competitive) goals I have identified only weigh further in favor of disclosure. Thus, one cluster of solutions involves protective orders, in-camera review, trade secret analysis by mutually-agreed-upon third-party experts or special masters, and other solutions.[558] In cases of extreme sensitivity, it is common for courts to issue protective orders limiting access to trade secrets only to counsel and their experts.[559] For example, in the election context, laws limit access to election officials or hold the code in escrow with an estab-

554 For a list of relevant questions and considerations, see Northern District of California's Model Protective Order for Litigation Involving Patents, Highly Sensitive Confidential Information, Northern District of California, *supra* note 553.

555 *See* Beckerman-Rodau, *supra* note 149, at 382.

556 In fact, the Rules of Practice for Patent Cases before the Eastern District of Texas require patent defendants to make the following available for inspection: "[s]ource code, specifications, schematics, flow charts, artwork, or other documentation sufficient to show the elements of an 'Accused Instrumentality.'" PATENT INITIAL DISCLOSURES 3-4, UNITED STATES DISTRICT COURT FOR THE EASTERN DISTRICT OF TEXAS, www.txed.uscourts.gov/?=patent-rules [https://perma.cc/955L-QDBT].

557 Sioux Pharm, Inc. v. Eagle Labs., Inc., 865 N.W.2d 528, 538 (Iowa 2015).

558 Chessman, *supra* note 43, at 213.

559 *See* Tailored Lighting, Inc. v. Osram Sylvania Prods., 236 F.R.D. 146, 148 (W.D.N.Y. 2006) (issuing protective order due to risk of economic injury); *see also* Seattle Times Co. v. Rhinehart, 467 U.S. 20, 36 (1984) ("The unique character of

lished third party and enable third parties to petition for access, thereby protecting the integrity of the system.[560] The Tenth Circuit recently observed that the disclosure of trade secrets on an "'attorneys' eyes only' basis is a routine feature of civil litigation involving trade secrets."[561] Processes like interposition allow for a trade secret to be revealed to a neutral third party who will inspect the trade secret in order to determine whether it is necessary to prove a case.[562] Another idea is to encourage courts to hold evidentiary, *in camera* hearings with expert testimony to determine whether the source code qualifies as a trade secret.[563] Expert testimony could be introduced to analyze the contents of the source code and to determine both whether it constitutes a trade secret and the parameters surrounding disclosure.[564]

For example, in a case involving algorithms, a district court upheld a detailed protective order, disclosing the source code information only to counsel and expert consultants and providing for additional security measures (such as the requirement that the information must be password protected, locked when not in use, and connected to a computer that cannot be connected to the internet).[565] The case—which is hardly unique—clearly shows that source code can be disclosed *and* protected, on a limited basis, in a judicial dispute. And a review of other cases suggests that courts have great acuity in addressing the issue.[566] In fact, in many conventional source code cases, it is

the discovery process requires that the trial court have substantial latitude to fashion protective orders.").

560 Gibson, *supra* note 20, at 190–91 (citing CAL. ELEC. CODE § 19205 (West 2003)).

561 Paycom Payroll, LLC. v. Richison, 758 F.3d 1198, 1202 (10th Cir. 2014) (quoting *In re* City of New York, 607 F.3d 923, 935 (2d Cir. 2010)). See for example, Pincheira v. Allstate Ins. Co., 190 P.3d 322, 333 (N.M. 2008) ("If the parties are not competitors, the trial court should issue an appropriate protective order and hold an evidentiary, adversarial hearing on the trade secret status of the information.").

562 26 CHARLES ALAN WRIGHT & ARTHUR R. MILLER, FEDERAL PRACTICE AND PROCEDURE § 5652, at 150 n.74 (1st ed.).

563 *See* Sea Coast Fire, Inc. v. Triangle Fire, Inc., 170 So. 3d 804, 808 (Fla. Dist. Ct. App. 2015); *see also* Hammock v. Hoffman-LaRoche, Inc., 635 A.2d 533, 538–39 (N.J. Super. Ct. App. Div. 1993), *rev'd*, 662 A.2d 546 (N.J. 1995) (noting difficulty with reviewing either judge's orders).

564 *See Sea Coast Fire*, 170 So. 3d at 808 (citing Revello Med. Mgmt., Inc. v. Med-Data Infotech USA, Inc., 50 So. 3d 678, 680 (Fla. Dist. Ct. App. 2010) (noting that if the judge is inexperienced in examining source code, he can appoint a neutral computer expert to review the program)).

565 Superior Edge, Inc. v. Monsanto Co., No. 12-2672 (JRT/FLN), 2014 WL 7183797, at *5 (D. Minn. Dec. 16, 2014).

566 The EPA statute, for example, allows a submitting company that has designated certain information as "trade secrets or commercial or financial informa-

important to note that source code has been turned over to authorities and still maintained its status as a trade secret.[567]

Finally, it bears noting that although the Federal Rules of Acquisition prohibit government employees from disclosing trade secrets, a number of other federal statutes extend permission to government agencies to disclose trade secret information when it is necessary to protect the public from harm to its safety and welfare.[568] The SEC, in addition, is governed by a statutory provision that gives it the authority to disclose trade secrets if it serves the public interest.[569] Even in the FOIA context, where trade secrets are granted an exemption, the Supreme Court has unanimously held that the exemption is discretionary for agencies, creating no mandatory bar to disclosure.[570] To take one example, the Honest and Open New EPA Science Treatment Act of 2017, in draft, would require the EPA to make documents that contained confidential business information available with redactions to the general public and without redactions to anyone who would sign a confidentiality agreement.[571]

In other words, the prospects for discovery and disclosure may be mixed, but there is some growing evidence to suggest that courts and legislators may be more willing to order source

tion" to institute a declaratory judgment action in federal district court if the company learns that the EPA plans to disclose that information. *See* Ruckelshaus v. Monsanto Co., 467 U.S. 986, 992 (1984) (citing Federal Environmental Pesticide Control Act of 1972, § 10(a), 86 Stat. 989 (1972)).

567 Courts have held that the taking of evidence of trade secrets can be done in camera, with no risk of violating the policy values that favor public trials. *See* State *ex rel.* Ampco Metal, Inc. v. O'Neill, 78 N.W.2d 921, 926–27 (Wis. 1956); House v. Commonwealth, No. 2007-CA-DG, 2008 Ky. App. Unpub. LEXIS 1220, at *19) (requiring source code disclosure in breathylzer case). *But see* State v. Kuhl, 741 N.W.2d 701, 708–09 (Neb. Ct. App. 2007) (reaching the opposite conclusion and deferring to trade secret protection).

568 *See* Stephen R. Wilson, *Public Disclosure Policies: Can a Company Still Protect Its Trade Secrets?*, 38 NEW ENG. L. REV. 265, 278 (2003) (mentioning statutes governing the Food and Drug Administration, Environmental Protection Agency, as examples.); *see also* Elizabeth A. Rowe, *Striking a Balance: When Should Trade-Secret Law Shield Disclosures to the Government?*, 96 IOWA L. REV. 791, 826–35 (2010) (addressing the circumstances under which the government can request disclosure).

569 Wilson, *supra* note 568, at 279.

570 *See* Chrysler Corp. v. Brown, 441 U.S. 281, 294 (1979) ("We therefore conclude that Congress did not limit an agency's discretion to disclose information when it enacted the FOIA."); Carol A. Ellingson, *The Copyright Exception for Derivative Works and the Scope of Utilization*, 56 IND. L.J. 1, 2–3 (1980) (discussing the derivative works exception). Interestingly, one wrinkle in such cases is that in some circumstances, trade secret holders have argued that they have a right to procedural due process, under *Mathews v. Eldridge*, 424 U.S. 319, 362 (1976) including an opportunity to be heard before the trade secret is disclosed.

571 *See* HONEST Act, H.R. 1430, 115th Cong. § 2 (2017).

code disclosure in justified cases. As Rebecca Wexler wryly observes, "disclosure subject to a protective order is better than no disclosure at all."[572] And review of source code, when it happens, can often mean a tremendous difference for due process and accountability, changing people's lives as a result.[573]

CONCLUSION

In *Lear v. Adkins*, the Supreme Court precipitously wrote, "federal law requires that all ideas in general circulation be dedicated to the common good unless they are protected by a valid patent."[574] Today, it is clear that trade secrecy's dominance over source code has been a significant cause for concern in cases involving the public interest. And, as I have shown, it is the failures of intellectual property law that have facilitated this result. To protect civil rights in the age of automated decision making, I argue, we must limit opportunities for seclusion in areas of intellectual property, criminal justice, and governance more generally. The solution, therefore, does not require a complete overhaul of the existing system, but rather a more nuanced, granular approach that seeks to balance the interest of disclosure and public access with the substantial values of protection, privacy, and property.

[572] Rebecca Wexler, *Life, Liberty, and Trade Secrets: Intellectual Property in the Criminal Justice System* 53 (Apr. 14, 2017) (unpublished draft).

[573] In one case, a review of Alcotest revealed that the source code had disabled catastrophic error detection, necessitating court intervention to secure its correction. *See* Ram, *supra* note 379, at 687 (citing State vs. Chun, 943 A.2d 114, 159 (N.J. 2008)). In another example from Colorado, programmers encoded over 900 errors in an algorithm that addressed the public benefit system; as a result, both cancer patients and pregnant individuals were wrongly denied Medicaid benefits, among other errors, costing the state several hundred millions of dollars, not to mention the individuals that were also directly affected. *See* Testimony of NYCLU, Hearing on Automated Decision Systems Used by Agencies, *supra* note 3, at 4 (citing Citron, *Technological Due Process*, *supra* note 490, at 1268–69) (follow "Hearing Testimony 10/16/17" hyperlink, page 44 of pdf).

[574] Lear, Inc. v. Adkins, 395 U.S. 653, 668 (1969).