Reverse Engineering and Trade Secrets in the Post-*Alice* World

*Samuel J. LaRoque**

I. INTRODUCTION

Consider a new software startup company, its employees working long hours to develop a software algorithm that will be bigger, better, and faster than the state-of-the-art already on the market. Maybe it will allow traders to respond to changing financial markets a millisecond faster than their competitors, or it will help airlines plan more efficient routes to save billions of dollars in fuel each year. The company believes this new algorithm will lead to an initial public offering worth tens of millions of dollars and early retirement. But before it releases the software and changes the world, it wants patent protection. It wants to prevent others from taking the same algorithm, packaging it with a slightly different product, and selling it for less because it had lower development costs. The company consults with several patent lawyers and receives discouraging news. In view of several recent Supreme Court decisions, most notably *Alice Corp. v. CLS Bank International*[1] in 2014, this algorithm will be nearly impossible to patent. This company cannot use patent protection to exclude others from using the algorithm, and the company's success is suddenly highly uncertain. What should the company do?

These are difficult times for software companies seeking to protect their intellectual property. The United States Supreme Court decided *Alice* in June 2014, holding that most computer software constitutes unpatentable abstract ideas.[2] *Alice* has therefore made software patents

* J.D. Candidate, 2018, University of Kansas School of Law; Ph. D., 2005, University of Chicago; B.A., 1998, Cornell University. The author thanks Professor Andrew Torrance for many helpful discussions and a careful examination of the manuscript. He also thanks Nathan Mannebach and Beth Hanus for invaluable input. Finally, he thanks Mathew Petersen, Adam Gillaspie, Mackenzie Sheehy, and the staff of the University of Kansas Law Review for their significant improvements to the manuscript.

1. 134 S. Ct. 2347 (2014).

2. *Alice*, 134 S. Ct. at 2357–59; note that while *Alice*'s holding was arguably narrow and limited to the claims at issue, it has since been widely interpreted by lower courts and the United

much more difficult to obtain and far easier to invalidate in the courts. This has driven software companies toward other forms of intellectual property protection for their software, including trade secrets. Trade secrets differ from patents, however, in at least one very significant way. Patent protection lasts for a fixed amount of time,[3] but once a trade secret is exposed, protection is lost forever.[4] One legitimate way to determine a trade secret is through reverse engineering, so software companies turning to trade secrets must carefully protect their products against reverse engineering by competitors or else risk losing protection.[5] Yet while patent protection for software faces more challenges than ever before, reverse engineering of software is becoming easier.[6] Software companies therefore face significant challenges in protecting their intellectual property, as both patent protection and trade secret protection look increasingly weak.

This Comment explores the extent to which software companies can go to prohibit reverse engineering of their marketed products in today's legal climate. As will be shown, while blanket prohibitions on reverse engineering will probably not succeed, software companies can likely use shrinkwrap agreements to ban reverse engineering for specific limited purposes such as direct economic competition.

Part II of this Comment provides important background information. Part II.A contains an overview of key software concepts needed to understand the legal analysis that follows. Part II.B contains a summary of the Supreme Court's recent *Alice* decision and its impact in restricting patentability for software. Part II.C begins with an overview of trade secret law and its application to software. Part II.C.1 provides an overview of reverse engineering as a proper means of discerning trade secrets, and Part II.C.2 contains background on reverse engineering of software.

Part III begins with a survey of recent results where courts have applied contract principles to intellectual property law. Part III.A examines how the courts have applied contract principles to copyright law specifically. Part III.B then examines the extension of these

States Patent and Trademark Office as a broad rebuke to software patents. *See infra* Section II.B.

3. Patents typically expire twenty years following the filing of the earliest related application, after which the protected invention is dedicated to the public. 35 U.S.C. § 154(a)(2) (2015 Supp.).

4. *E.g.*, 1 ROGER M. MILGRIM, MILGRIM ON TRADE SECRETS § 1.05(1), (2) (2017) [hereinafter MILGRIM] (describing loss of secrecy and subsequent termination of trade secret protection through sale, display, or circularization).

5. *Id.* at § 1.05(5).

6. *See* discussion *infra* Section II.C.2.

copyright precedents to trade secret law, challenges posed by existing patent law, and how software companies might apply contract principles to protect their trade secrets. Part III.B.1 examines whether a total contractual ban on reverse engineering to discern trade secrets embedded in a software company's product could succeed. Part III.B.2 looks at the more reasonable option of a limited contractual ban, one that prohibits reverse engineering only for certain purposes. Both case law (Part III.B.2.i) and statutory (Part III.B.2.ii) support for limited contractual bans are discussed.

II. BACKGROUND

A. Software, Generally

A basic understanding of several major concepts in software is useful to understand recent developments in software patentability, as well as the benefits and challenges to protecting software with trade secrets. The term "software" refers generally to the programs that direct the activities on a computer system.[7] While there are many kinds of software, the focus here is on so-called "applications software." Applications software controls particular applications such as word processing, photo editing, e-mail management and so on. These are the kinds of products that consumers tend to buy "off-the-shelf"[8] to install on their computers to perform certain tasks.

Software resides in a computer's internal memory, and is generally placed there by transferring it from a disk or downloading it from a source such as the Cloud or the World Wide Web. The stored software consists of instructions that tell the computer (specifically the computer's processing unit) what to do. Examples of basic instructions include adding two numbers, or moving a value from one memory location to another. A typical application program is made up of many millions of such instructions.

Ultimately humans write software to tell computers what to do through the process of programming. Programming fundamentally "involves writing a program in some code or language that the computer

7. *See generally* RONALD J. TOCCI & FRANK J. AMBROSIO, MICROPROCESSORS AND MICROCOMPUTERS: HARDWARE AND SOFTWARE 169 (6th ed. 2003) (providing a general overview of the types of software programs).

8. Not to be taken literally, purchasing software "off-the-shelf" is understood to include downloading software from a website, which is now a common method of obtaining software.

can [understand]."[9] Today's computers are digital, rather than analog, devices. Digital devices are designed to operate using voltage signals, which can have only two possible values: low voltage, represented by a zero, and high voltage, represented by a one.[10] Human programmers can write programs consisting of instructions made up of ones and zeroes, but doing so is tedious and errors are nearly impossible to detect and fix.[11]

Programmers therefore generally write computer programs in high-level languages such as C++, Java, or Visual Basic.[12] Such languages make programmers' task much easier by using English-language words and combining many individual computer instructions into a single statement.[13] Computers themselves do not understand the high-level language, however, so it must be translated into the strings of ones and zeroes that computers recognize. For higher-performance applications, this is done through the process of compilation. Compilers are themselves computer programs which take each high-level statement and translate it into the series of ones and zeroes that a particular computer understands.[14] This new, translated program—sequences of ones and zeroes stored in the computer's memory—is called an "object program" consisting of "object code."[15]

Because high-level languages use English-language constructs, a programmer should be able to understand the functionality of another programmer's work simply by reviewing the high-level code. By contrast, even a highly-experienced programmer probably would not understand the meaning of thousands or millions of strings of ones and zeroes—the object code—without a great deal of painstaking analysis. Microsoft Word, for instance, is written by programmers in the high-level C++ language.[16] Anyone with even a basic knowledge of programming would recognize much of the basic functionality of Word at this level. Even the most experienced programmer, however, would find the object code virtually impenetrable. The detailed programming methods used to solve the various problems that Word presents would be

9. TOCCI & AMBROSIO, *supra* note 7, at 170.
10. *See, e.g.*, *id.* at 35–36.
11. *Id.* at 170–71.
12. *Id.* at 174.
13. *Id.*
14. *Id.* at 175.
15. *See id.* at 170.
16. *See, e.g.*, *The Programming Languages Beacon*, LEXTRAIT.COM (Mar. 5, 2016), http://www.lextrait.com/vincent/implementations.html (graphically indicating that Microsoft Office, which includes Word, is developed using C++).

almost totally obscured. As will be shown, selling already-compiled code in object code format is a valuable way for software companies to protect the secrecy of their specific approaches to problem solving.[17] This has long been an alternative to patent protection, and is as important today as ever before in view of the Supreme Court's decision in *Alice*.

B. The Impact of Alice

The legal landscape for patenting software became much more difficult in 2014 when the United States Supreme Court handed down its decision in *Alice Corp. v. CLS Bank International*.[18] The Court in *Alice* established a much more stringent test for patentability that excludes many software algorithms. Such algorithms often create competitive advantages in the marketplace, as when a company's new algorithm leads to improvements in speed or memory use compared with its competitors.[19] These advantages can be valuable intellectual property rights, but the patentability of many such algorithms is now highly questionable in the wake of *Alice*.

The status of software patents prior to *Alice* was murky, but several major cases had suggested that software was generally patentable. For instance, the Federal Circuit signaled its acceptance of business method patents in *State Street Bank & Trust Co. v. Signature Financial Group, Inc.* in 1998.[20] In doing so, it held that a computational method for transforming data into a final share price using a series of mathematical calculations was patent-eligible subject matter because that final share price was a "useful, concrete, and tangible result."[21] The Federal Circuit narrowed its view in *In re Bilski* in 2008, finding that the "useful, concrete, and tangible result" test on its own was insufficient for demonstrating patentability.[22] It relied instead on the Supreme Court's "machine-or-transformation" test, noting that when a process was not tied to a particular machine or apparatus (e.g., an algorithm to be run on a generic computer) the test for patentability was simply whether it

17. *See infra* note 59 and accompanying text.

18. Alice Corp. v. CLS Bank Int'l, 134 S. Ct. 2347 (2014).

19. Relevancy is another example, as best illustrated by Google—its success stemmed largely from the fact that its PageRank algorithm returned more relevant results than the algorithms of its competitors. *See* John Battelle, *The Birth of Google*, WIRED (Aug. 1, 2005, 12:00 PM), https://www.wired.com/2005/08/battelle/?tw=wn_tophead_4.

20. State St. Bank & Tr. Co. v. Signature Fin. Grp., Inc., 149 F.3d 1368 (Fed. Cir. 1998).

21. *Id.* at 1375 (quoting *In re* Alappat, 33 F.3d 1526, 1544 (Fed. Cir. 1994)).

22. *In re* Bilski, 545 F.3d 943, 959–60 (Fed. Cir. 2008), *aff'd sub nom.* Bilski v. Kappos, 561 U.S. 593 (2010).

transformed an article into a different state or thing.[23] It further held that transformations of data, such as those that might be carried out by a software algorithm, were patentable under this test.[24] Even when the Supreme Court scaled back patent protection for business methods in *Bilski v. Kappos*, finding that the patent statutes "[did] not suggest broad patentability of such claimed inventions," the Court did not go so far as to restrict patentability for software.[25] Rather, it referred to the machine-or-transformation test as a "useful . . . investigative tool" for evaluating patent eligibility.[26]

Alice changed all of this, as courts have generally interpreted the opinion as a major rebuke to software patents.[27] *Alice* involved patent claims directed to mitigating so-called settlement risk, which is the risk that only one party to a financial exchange satisfies its obligation.[28] The claims recited a computer system for exchanging financial obligations, including software for performing the exchange.[29] In analyzing these claims, the Court relied on the two-part patentability test it set forth in *Mayo v. Prometheus Labs*.[30] Under this test, one first determines whether the claims are directed to a patent-ineligible concept.[31] If so, one then determines whether the elements of the claim transform it into a patent-eligible application.[32] Applying the first part of the test, the Court found that intermediated settlement by a third party is an abstract idea—a patent-ineligible concept.[33] It then moved on to the second part of the test. In applying the second part, the Court found that the patent claims were directed only to a generic computer implementation of intermediated risk, and therefore failed to transform the abstract idea into a patent-eligible invention.[34]

23. *Id.* at 962.

24. *See id.* at 962–63 (describing how transforming raw data into a visual depiction of a tangible object, as in CT scanning, was patent-eligible).

25. *Bilski v. Kappos*, 561 U.S. at 608.

26. *Id.* at 604.

27. *See infra* notes 35–40 and accompanying text (discussing software patent challenges post-*Alice*).

28. Alice Corp. v. CLS Bank Int'l, 134 S. Ct. 2347, 2349 (2014).

29. *Id.*

30. *Id.* at 2355 (citing Mayo Collaborative Servs. v. Prometheus Labs., Inc., 132 S. Ct. 1289 (2012)).

31. *Id.* (noting that patent ineligible subject matter includes "laws of nature, natural phenomena, and abstract ideas").

32. *Id.*

33. *Id.* at 2356–57.

34. *Id.* at 2360.

Although its ineligibility finding was limited to the claims at hand, courts have thus far interpreted *Alice*'s holding as a broad rebuke of software patents in general. Within roughly two years of the *Alice* decision, 568 patents had been challenged on patentability grounds through motions citing *Alice* at the Patent Trial and Appeals Board (PTAB), district courts, and the Federal Circuit.[35] Overall, these judicial bodies applied *Alice* to invalidate 378 patents and patent applications out of 568 considered in the two years following the decision, an invalidation rate of sixty-seven percent.[36] The Federal Circuit has been particularly harsh on software patents, invalidating thirty-four of the thirty-seven software patents it examined under *Alice* within two years of the decision.[37] Meanwhile, by June 2016 the USPTO had rejected more than 36,000 patent applications over *Alice*, of which patent applicants had abandoned[38] more than 5,000.[39] Statistics from the USPTO further show that rejection rates for software-related patents spiked after *Alice* and remained high well into 2017.[40]

Despite its force in invalidating software patents, the scope of *Alice*'s holding remains unclear. The Supreme Court did not consider *Alice* to be a case about software patents, and the parties to the case agreed that Alice Corporation itself had never actually written software.[41] It is the broad interpretation of *Alice* by both the Federal Circuit and lower courts—which have thus far interpreted it as a nearly per se rule against software patents—that has had such a profound effect.[42] Despite its substantial impact, *Alice* is a short case (the Supreme Court considered it "minor"),[43] and the Court spent very little time describing how to apply

35. Jasper L. Tran, *Two Years After Alice v. CLS Bank*, 98 J. OF THE PAT. & TRADEMARK OFF. SOC'Y 354, 358 (2016).

36. *Id.*

37. *Id.*

38. To abandon a patent application means to cease pursuing a patent based on that application.

39. Tran, *supra* note 35, at 358–59 (citing Robert R. Sachs, *Two Years After Alice: A Survey of the Impact of a "Minor Case" (Part 2)*, FENWICK & WEST'S BILSKI BLOG (June 20, 2016), http://www.bilskiblog.com/blog/2016/06/two-years-after-alice-a-survey-of-the-impact-of-a-minor-case-part-2.html.

40. Robert R. Sachs, *#Alicestorm: April Update and the Impact of TC Heartland on Patent Eligibility*, FENWICK & WEST'S BILSKI BLOG (June 1, 2017), http://www.bilskiblog.com/blog/2017/06/alicestorm-april-update-and-the-impact-of-tc-heartland.html.

41. Robert R. Sachs, *Two Years After Alice: A Survey of the Impact of a "Minor Case" (Part 1)*, FENWICK & WEST'S BILSKI BLOG (June 16, 2016), http://www.bilskiblog.com/blog/2016/06/two-years-after-alice-a-survey-of-the-impact-of-a-minor-case.html.

42. *Id.*

43. *Id.*

each prong of its test.[44] This is of no consolation to software companies, however, who may have difficulty protecting their algorithms and finding funding for new software ventures in the wake of so much uncertainty.[45]

Alice does not appear to spell the end of software patents altogether. The Court seemed to leave open the possibility that claims that improve the functioning of the computer itself, or that improve on an existing technological process, are still patent eligible.[46] The Federal Circuit has also recently upheld software patents in several high-profile cases.[47] But these cases are the exception rather than the rule. Both the federal district courts and the Federal Circuit have repeatedly struck down software patents as non-patentable subject matter in the wake of *Alice*, finding that the claims are directed to abstract ideas.[48] With the Federal Circuit upholding just eight software patents in the more than three years since *Alice*,[49] software companies are struggling to find other ways of protecting their intellectual property.

Some software companies appear to be turning to copyright protection for their programs in the wake of *Alice*. Congress amended 17 U.S.C. § 101 in 1980 to expressly make software eligible for copyright,[50] so there seems to be little risk that the courts will clamp down on software copyrights as they have with patents. There are advantages to copyrighting software, and in many instances this may be the preferred

---

44. *See, e.g.*, Alice Corp. v. CLS Bank Int'l, 134 S. Ct. 2347, 2357 (2014) ("[W]e need not labor to delimit the precise contours of the 'abstract ideas' category in this case. It is enough to recognize that there is no meaningful distinction between the concept of risk hedging in *Bilski* and the concept of intermediated settlement at issue here.").

45. *See, e.g.*, *Alice, Abstract Ideas, and Software-Related Patents*, BERKELEY TECH. L. J.: BTLJ BLOG (Mar. 1, 2016), http://btlj.org/2016/03/alice-abstract-ideas-software-related-patents/.

46. *See Alice*, 134 S. Ct. at 2359–60 (noting that the claims were invalid in part because they did not "purport to improve the functioning of the computer itself or effect an improvement in any other technology or technical field").

47. *See, e.g.*, Bascom Glob. Internet Servs., Inc. v. AT&T Mobility LLC, 827 F.3d 1341, 1350–51 (Fed. Cir. 2016) (holding that although claims directed to filtering of internet content were directed to an abstract idea, the claims improved on an existing technological process and therefore did not monopolize the abstract idea); Enfish, LLC v. Microsoft Corp., 822 F.3d 1327, 1336 (Fed. Cir. 2016) (upholding claims directed to a "self-referential" database table upon finding that they were directed to improving the functionality of the computer itself); DDR Holdings, LLC v. Hotels.com, L.P., 773 F.3d 1245, 1259 (Fed. Cir. 2014) (upholding claims directed to a web server on the basis that they were directed to a specific "Internet-centric problem" rather than an abstract idea).

48. *See supra* notes 35–37 and accompanying text.

49. Sachs, *supra* note 40.

50. *See, e.g.*, Lawrence D. Graham & Richard O. Zerbe, Jr., *Economically Efficient Treatment of Computer Software: Reverse Engineering, Protection, and Disclosure*, 22 RUTGERS COMPUTER & TECH. L.J. 61, 91 (1996).

method of IP protection. Such an analysis is beyond the scope of this Comment. Yet copyright protection for software has at least two significant weaknesses. First, one recent court decision appears to significantly expand fair use protections for copying software.[51] Second, copyright protection in software is limited to an author's original creations. This means that coding constructs that are highly generic, or already in the public domain, do not receive protection and are therefore subject to copying by others.[52] As a result, where generic constructs can be used to code specific algorithms, copyright protection might actually protect very little.

The other main alternative to patents is trade secrets. *Alice* has not altered trade secret protection in any way. As this Comment will show, trade secrets can therefore be a valuable means for protecting software if used correctly.

C. *The Trade Secret (Plus Contract) Solution*

In view of tighter restrictions on software patents in the wake of *Alice*, software companies are increasingly turning to trade secrets to protect their valuable algorithms.[53] But in doing so, such companies should consider the permissive posture of existing trade secret law toward reverse engineering. Competitors, after all, may use reverse engineering to compromise the protections that software owners seek.

Statutory trade secret protections have evolved substantially over the past eighty years. Trade secret law was initially compiled in the first Restatement of Torts in 1939.[54] But the primary source of trade secret law in most jurisdictions today is the Uniform Trade Secrets Act (UTSA), first promulgated in 1979 and now adopted by forty-seven

51. *See* Oracle Am., Inc. v. Google Inc., No. C 10-03561, 2016 U.S. Dist. LEXIS 145601 (N.D. Cal. Sept. 27, 2016) (reiterating jury finding that Google's copying of Oracle's Java API was protected by fair use exceptions under copyright law), *appeal docketed and consolidated with* No. 17-1118 (Fed. Cir. Nov. 14, 2016).

52. *See, e.g.*, Comput. Assocs. Int'l, Inc. v. Altai, Inc., 982 F.2d 693, 714–15 (2d Cir. 1992) (noting that "elements taken from the public domain do not qualify for copyright protection" and finding no copyright violation).

53. *See, e.g.*, Ryan Davis, *Attorneys Lean Toward Trade Secrets to Avoid* Alice *Headaches*, LAW360 (July 17, 2015, 3:54 PM), https://www.law360.com/articles/666273/attys-lean-toward-trade-secrets-to-avoid-alice-headaches (describing general strategy shift toward embracing trade secrets for software in view of *Alice*); Stephanie Forshee, *In Fintech, Trade Secrets May Be Replacing Patent Applications*, CORPORATECOUNSEL (June 21, 2017), http://www.law.com/corpcounsel/almID/1202790865489/ (describing a shift away from patent applications and toward trade secrets in the financial technology industry and attributing it to *Alice*).

54. *See* MILGRIM, *supra* note 4, § 1.01(2)(c)(i) (2017).

states and the District of Columbia.[55] The UTSA defines a trade secret as information, including a program, method, or process, that "derives independent economic value . . . from not being generally known to . . . [or] readily ascertainable by" other persons using proper means.[56] Those "other persons" must further benefit in some way from use of the trade secret.[57] The UTSA further requires that trade secret owners make "efforts . . . reasonable under the circumstances to maintain [this] secrecy."[58] Computer software is well suited to these UTSA requirements. First, the UTSA definition expressly encompasses programs, and extends also to methods and processes which are often embodied in software algorithms. Second, distribution of software in object code format generally meets the basic requirement of maintaining secrecy because it is not easily decipherable to human readers.[59]

The UTSA defines trade secret misappropriation generally as the acquisition of another's trade secret by one who knows or had reason to know the acquisition was by improper means, or the disclosure or use of another's trade secret without express or implied consent where the trade secret was acquired through improper means.[60] "Improper means" generally refers to theft, bribery, misrepresentation, and even legal actions that fall below generally accepted standards of commercial conduct.[61]

Trade secret protection historically existed almost entirely at the state level until Congress enacted the federal Defend Trade Secrets Act (DTSA) in May 2016.[62] The DTSA provides a federal cause of action for trade secret misappropriation.[63] It closely mirrors the UTSA, and the two acts define "trade secret" and "misappropriation" almost

55. *Id.* at § 1.01(2)(b).

56. UNIF. TRADE SECRETS ACT WITH 1985 AMENDMENTS § 1(4)(i) (UNIF. LAW COMM'N 1985).

57. *See id.*

58. *Id.* § 1(4)(ii).

59. *See, e.g.*, Data Gen. Corp. v. Grumman Sys. Support Corp., 825 F. Supp. 340, 359 (D. Mass. 1993) ("Even those who obtained [the software] and were able to use [it] were unable to discover its trade secrets because [the software] was distributed only in its object code form, which is essentially unintelligible to humans.").

60. *See* UNIF. TRADE SECRETS ACT § 1(2).

61. *Id.* at § 1(1).

62. Defend Trade Secrets Act of 2016, Pub. L. No. 114-153, § 2(f), 130 Stat. 376, 382 (2016) (codified at 18 U.S.C. §§ 1836–1839).

63. MARK L. KROTOSKI ET AL., MORGAN, LEWIS & BOCKIUS LLP, THE LANDMARK DEFEND TRADE SECRETS ACT OF 2016 at 7 (2016), https://www.morganlewis.com/~/media/files/publication/morgan%20lewis%20title/white%20paper/the-landmark-defend-trade-secrets-act-of-2016-may2016.ashx?la=en.

identically.[64] Under the DTSA, a trade secret owner can bring an action in federal court if the trade secret is related to a product or service used in interstate or foreign commerce.[65] Notably, the DTSA expressly states that it does not preempt state trade secret law.[66] Trade secret owners can therefore choose whether to bring an action in state court under state law or in federal court under the DTSA.[67]

1. Reverse Engineering in Trade Secret Law

The issue of reverse engineering has a long history in trade secret law. Reverse engineering is defined as "starting with the known product and working backward to find the method by which it was developed,"[68] and courts have traditionally recognized reverse engineering as a proper means of learning trade secrets. In *Kewanee Oil Co. v. Bicron Corp.*, for instance, the United States Supreme Court explained that trade secret law does not protect against discoveries by fair and honest means including "so-called reverse engineering, that is by starting with the known product and working backward to divine the process which aided in its development or manufacture."[69]

A more detailed treatment of reverse engineering comes from *Chicago Lock Co. v. Fanberg*, where plaintiff lock company sued defendant locksmiths for trade secret misappropriation when the defendants compiled a list of key codes for the plaintiff's locks by contacting other locksmiths who had already picked the locks.[70] The court held not only that the reverse engineering by individual locksmiths was not misappropriation, but that the sharing of the key codes with other locksmiths was not misappropriation either.[71] The court concluded by noting that to hold otherwise would bring state trade secret protection into line with the absolute monopoly afforded by patents and lead to preemption by federal patent law.[72]

The UTSA does not expressly address reverse engineering in its statutory language, but the Comments state that reverse engineering is an

64. *Compare* 18 U.S.C.A. § 1839(3), (5) (West Supp. 2016), *with* UNIF. TRADE SECRETS ACT § 1(1), (2).

65. 18 U.S.C.A. § 1836(b)(1) (West Supp. 2016).

66. 18 U.S.C.A. § 1838 (West Supp. 2016).

67. KROTOSKI ET AL., *supra* note 63, at 7–8.

68. UNIF. TRADE SECRETS ACT § 1 cmt.

69. Kewanee Oil Co. v. Bicron Corp., 416 U.S. 470, 476 (1974).

70. Chi. Lock Co. v. Fanberg, 676 F.2d 400, 402–03 (9th Cir. 1982).

71. *Id.* at 405.

72. *Id.*

appropriate means of discerning a trade secret. Section 1 of the UTSA defines "trade secret" in terms of a requirement that the secret be "not . . . readily ascertainable by proper means."[73] The comments to the statute include the following statement:

> Proper means include: . . . 2. Discovery by "reverse engineering", that is, by starting with the known product and working backward to find the method by which it was developed. The acquisition of the known product must, of course, also be by a fair and honest means, such as purchase of the item on the open market for reverse engineering to be lawful . . . .[74]

The federal DTSA takes the UTSA one step further by expressly addressing reverse engineering in the statutory language. After defining "misappropriation" in terms of acquisition or disclosure using improper means, the DTSA defines "improper means" as including "theft, bribery, misrepresentation, breach or inducement of a breach of a duty to maintain secrecy, or espionage through electronic or other means."[75] It then expressly excludes reverse engineering from "improper means," stating that the term "improper means" does not include "reverse engineering, independent derivation, or any other lawful means of acquisition . . . ."[76] The trend is therefore clear: the courts have long held that reverse engineering is a legitimate means of acquiring trade secrets, the UTSA recognized this when defining trade secrets, and the federal government has recently emphasized this in the DTSA.

2. Reverse Engineering of Software

Computer software is unique in that it cannot simply be taken apart and examined physically like the locks in *Chicago Lock*. As discussed earlier, software companies typically compile their source code into object code prior to distributing it.[77] Courts have generally held that distribution of this object code reflects a sufficient effort to maintain secrecy because unlike the original source code, this object code is indecipherable to human readers.[78] Object code, however, can be reverse

---

73. UNIF. TRADE SECRETS ACT § 1(4)(i).
74. *Id.* § 1 cmt.
75. 18 U.S.C.A. § 1839(6)(A) (West Supp. 2016).
76. *Id.* § 1839(6)(B).
77. *See supra* Section II.A.
78. *See, e.g.*, Trandes Corp. v. Guy F. Atkinson Co., 996 F.2d 655, 663–64, 663 n.8 (4th Cir. 1993) (determining that the object code at issue met the definitional requirements for a trade secret);

engineered through computational processes to produce a rough copy of the algorithms in the original source code; this process is often referred to as "disassembly" or "decompilation."[79] While the reproduction is far from exact, this rough copy may be sufficient for the reverse engineering party to discern what makes the software faster or more efficient. A software company might have patented such information prior to *Alice*, and now might wish to protect it as a trade secret. If the information is discerned through reverse engineering, however, trade secret protection is no longer available and the software company may lose a valuable edge on its competitors.

Past scholarship has largely dismissed decompilation as a legitimate threat to software trade secrets owing to the time, effort, and expense required.[80] But it is time to reexamine the risks of decompilation in view of recent technological advances. As an example, consider the Java programming language. Like C or C++, Java is a high-level programming language that must be compiled (translated) into something the computer's processor understands before the code is run on that computer. Unlike C and C++, however, Java is compiled into an intermediate version of object code called Java bytecode prior to distribution over the web.[81] Final compilation occurs on any machine running software called the Java Virtual Machine (JVM).[82] Java bytecode is platform independent, meaning that the same high-level Java source code can be partially compiled into Java bytecode and distributed to many different types of machines, each of which then finishes the compilation process.[83]

This platform-independence makes Java very flexible and therefore a favorite for web-related applications because the same code can run on many different types of machines. But it also makes Java particularly vulnerable to reverse engineering.[84] The same tools readily available to

---

Q-Co Indus. v. Hoffman, 625 F. Supp. 608, 617 (S.D.N.Y. 1985) ("Only the object code is publically [sic] available; this [is] the version of the program that is intended to be read by the computer and cannot be understood even by expert programmers.").

79. *See, e.g.*, Pamela Samuelson & Suzanne Scotchmer, *The Law and Economics of Reverse Engineering*, 111 YALE L.J. 1575, 1608–09 (2002) (explaining how engineers can use disassembly or decompilation to "discern or deduce internal design details of the program").

80. *See, e.g.*, *id.* at 1613–14.

81. Stephen Rauh, *A Java Programmer's Guide to Byte Code*, BEYOND JAVA (Jan. 5, 2015), https://www.beyondjava.net/blog/java-programmers-guide-java-byte-code/.

82. *See id.*

83. *See id.*

84. *See, e.g.*, Ajay Yadav, *Java Bytecode Reverse Engineering*, INFOSEC INST.(Jan. 31, 2014), http://resources.infosecinstitute.com/java-bytecode-reverse-engineering ("It is relatively easy to disassemble the bytecode of a Java application, compared to other binaries.").

help Java users run Java bytecode through the JVM can also be used for disassembly as part of the reverse engineering process. Disassembly is, after all, essentially the opposite of compilation. And because Java end-users are closer to the compilation process, they have easier access to disassembly tools. The Eclipse integrated development environment (IDE) is one example of freely downloadable software that can be used to disassemble Java bytecode and potentially reverse engineer web applications software.[85]

While the object code associated with other high-level languages such as C or C++ tends to be more resistant to reverse engineering than Java, sophisticated disassembly tools exist for these languages as well, and skilled software engineers may be able to reproduce the basic functionality of a program written in these languages. Snowman, for instance, is a new C/C++ decompiler that can be used to translate object code into human-readable source code.[86] As with many decompilers, Snowman's developers advertise the product toward computer virus analysts as well as persons who have simply lost their source code.[87] But the product is not limited to these uses, and competitors could just as easily use it to discern a software company's protected algorithms through reverse engineering.

Software companies seeking to protect their intellectual property therefore face a major challenge—just as heightened patentability standards push them toward trade secret protection, advances in reverse engineering of software make it more difficult to protect their trade secrets. And these companies cannot rely on trade secret law to justify such bans. If intellectual property law alone won't sufficiently protect software companies looking to protect their investments post-*Alice*, then where should they turn? The answer, as argued here, may be contract law as applied to trade secrets.

III. ANALYSIS

When the courts have considered intellectual property protection in the context of contract law, rather than intellectual property law, the results have sometimes been more favorable to the holders of the

85. *Download Eclipse Technology That Is Right for You*, ECLIPSE, https://www.eclipse.org/downloads/ (last visited Oct. 17, 2017).

86. *Snowman*, DEREVENETS.COM (June 6, 2017, 11:14 PM), http://derevenets.com/.

87. Paul Krill, *C/C++ Decompiler Translates Programs, No Source Code Needed*, INFOWORLD (Oct. 14, 2014), http://www.infoworld.com/article/2833714/c-plus-plus/snowman-seeks-to-be-llvm-for-decompilers.html.

intellectual property rights. It was not always that way. In 1988, in *Vault Corp. v. Quaid Software Ltd.*, the Fifth Circuit invalidated a Louisiana statute that allowed shrinkwrap licenses[88] to prohibit reverse engineering of software for adaptability purposes.[89] The Fifth Circuit held that the provision in question "'touche[d] upon an area' of federal copyright law" and was therefore preempted by the federal Copyright Act.[90] This marked a significant setback for holders of intellectual property rights in software.

Although the next notable case did not involve a statute or reverse engineering, and so was not directly relevant to the holding in *Vault*, *ProCD, Inc. v. Zeidenberg* raised the hopes of software companies by holding that shrinkwrap licenses were not per se invalid.[91] There the Seventh Circuit upheld a shrinkwrap license on a software box instructing purchasers that they were bound by the terms of a license inside the box, including a term barring non-commercial uses of the software.[92] Building on *ProCD*, two more recent cases directly disagreed with *Vault* by upholding similar prohibitions against reverse engineering in shrinkwrap software licenses. In 2003, in *Bowers v. Baystate Technologies, Inc.*, the Federal Circuit relied on *ProCD* in upholding a shrinkwrap provision prohibiting reverse engineering.[93] And in 2005, in *Davidson & Associates v. Jung*, the Eighth Circuit closely followed *Bowers* in upholding a similar shrinkwrap provision prohibiting reverse engineering for interoperability purposes.[94]

The common theme of shrinkwrap licenses across these cases is no coincidence. As the cases demonstrate, software companies have tried to circumvent historically permissive policies toward reverse engineering software by relying instead on contract principles. Meanwhile, the issue of reverse engineering software has heightened relevance today. As software companies in an unfriendly patent landscape increasingly turn to trade secrets to protect their algorithms, they must deal with the very real possibility that competitors in the industry will reverse engineer their products to reveal their trade secrets and use those secrets to compete

88. A shrinkwrap license is a message printed on the outside of software packaging notifying users that they will be bound by license terms once they open the package. More common today are electronic versions requiring users to click a button on their computer or handheld screen to agree to terms; these are generally referred to as point-and-click agreements or clickwrap licenses.

89. Vault Corp. v. Quaid Software Ltd., 847 F.2d 255, 270 (5th Cir. 1988).

90. *Id.*

91. ProCD, Inc. v. Zeidenberg, 86 F.3d 1447, 1455 (7th Cir. 1996).

92. *Id.* at 1448–50.

93. Bowers v. Baystate Techs., Inc., 320 F.3d 1317, 1325 (Fed. Cir. 2003).

94. Davidson & Assocs. v. Jung, 422 F.3d 630, 639 (8th Cir. 2005).

directly. The courts seem loath to restrict reverse engineering and the DTSA reflects this permissive approach.

Shrinkwrap license provisions prohibiting reverse engineering for certain purposes offer another path. The law is unsettled here, as the cases described above demonstrate. The Fifth Circuit has held that shrinkwrap provisions prohibiting reverse engineering are invalid, yet the Federal Circuit and Eighth Circuit appear to allow them. Furthermore, those cases examined the relationship between shrinkwrap prohibitions on reverse engineering and copyright, not trade secrets. The key question, therefore, is just how far software companies can push these shrinkwrap licenses to protect their trade secrets from reverse engineering.

A. *Case Law Applying Contract Principles to the Protection of Intellectual Property*

To understand the extent to which software companies might rely on contract principles to protect their trade secrets, it is useful to first review the key cases in which software owners have used shrinkwrap terms—successfully or not—to protect their products.

1. *Vault v. Quaid*

In *Vault v. Quaid*, the Fifth Circuit held that a Louisiana licensing statute permitting shrinkwrap licenses to restrict reverse engineering was preempted by federal copyright law.[95] Vault produced computer diskettes containing copyrighted code to prevent unauthorized duplication of other companies' software programs.[96] Vault included a license agreement with each software package prohibiting "copying, modification, translation, decompilation or disassembly of [its] program."[97] Quaid produced and sold diskettes containing a feature specifically designed to unlock Vault's protection.[98] Quaid did not dispute that it had developed its own program by reverse engineering Vault's,[99] which violated the terms of the shrinkwrap agreement.

Vault sued, alleging that Quaid breached the license agreement by reverse engineering Vault's software in violation of the Louisiana

95. *Vault*, 847 F.2d at 270.
96. *Id.* at 256.
97. *Id.* at 257.
98. *Id.*
99. *Id.*

Software License Enforcement Act (LSLEA).[100] Under the LSLEA, software producers such as Vault could impose certain contractual terms on purchasers so long as the license agreement accompanied the producer's software.[101] Among those terms was one prohibiting "adaptation by reverse engineering, decompilation or disassembly."[102]

On appeal, the Fifth Circuit found the statute invalid and the license therefore unenforceable, holding that certain provisions of the statute "'touched upon the area' of federal copyright law" and were therefore preempted by it.[103] The Fifth Circuit specifically noted that § 117 of the Copyright Act permits an owner of a computer program to adapt that program for certain purposes.[104] The court held that the LSLEA provision prohibiting adaptation by reverse engineering touched upon this particular area of federal copyright law and was preempted by it.[105]

2. *ProCD v. Zeidenberg*

While *Vault* examined a specific provision in a software license, the seminal case examining enforceability of software shrinkwrap licenses in other contexts is *ProCD, Inc. v. Zeidenberg*.[106] There the Seventh Circuit held that consumers purchasing off-the-shelf software products were bound by the terms of the license agreements inside.[107] ProCD compiled listings from various telephone books into a computer database and sold the software to both commercial and private consumers so they could look up listings by simply querying the database.[108] It placed a message on every consumer software box stating that the buyer was bound by restrictions in an enclosed license, including a prohibition on using the application for non-commercial purposes.[109] But Matthew Zeidenberg did just that—he bought a consumer package and ignored the enclosed license terms, setting up a corporation to resell ProCD's listing

100. *Id.* at 268.
101. *Id.*
102. *Id.* at 268–69.
103. *Id.* at 269–70.
104. *Id.*
105. *Id.*
106. 86 F.3d 1447 (7th Cir. 1996).
107. *Id.* at 1448–49.
108. *Id.* at 1449.
109. *Id*. at 1450.

information to other companies for a profit.[110] ProCD sued, claiming that Zeidenberg had violated the license terms.[111]

On appeal, the Seventh Circuit upheld the license on the basic contract principle of mutual assent, holding that the Uniform Commercial Code did not preclude binding a purchaser to license terms hidden within packaging.[112] More relevant to the trade secret issue, however, was the court's examination of preemption. The district court concluded that the shrinkwrap license was unenforceable in view of § 301(a) of the Copyright Act,[113] which states in part that:

> [A]ll legal or equitable rights that are equivalent to any of the exclusive rights within the general scope of copyright as specified by section 106 in works of authorship that are fixed in a tangible medium of expression and come within the subject matter of copyright . . . are governed *exclusively* by this title.[114]

The Seventh Circuit reversed, holding that rights created by contract are not "equivalent to any of the exclusive rights within the general scope of copyright," and that there was therefore no preemption of state contract laws.[115] It reasoned that the exclusive rights described in the Copyright Act are rights against the world—all persons are forbidden from copying copyrighted material, whether they agree to it or not.[116] Contract rights, by contrast, affect only the parties to the contract and are therefore not "exclusive."[117] As such, shrinkwrap terms restricting certain uses of enclosed software were enforceable.[118]

3. *Bowers v. Baystate*

Although the Fifth Circuit in *Vault* held that a license provision prohibiting certain kinds of reverse engineering was unenforceable, the momentum has ultimately shifted from licensees back toward licensors on the issue of reverse engineering. *Bowers* marked the start of that shift. In *Bowers*, the Federal Circuit held that federal copyright law did

110. *Id.*
111. *Id.*
112. *Id.* at 1452–53.
113. *Id.* at 1454.
114. 17 U.S.C. § 301(a) (2012) (emphasis added).
115. *ProCD*, 86 F.3d at 1454 (quoting 17 U.S.C. § 301(a) (2012)).
116. *Id.*
117. *Id.*
118. *See id.* at 1455.

not preempt shrinkwrap license terms prohibiting reverse engineering.[119] Harold Bowers invented an improvement for computer aided design (CAD) software, which he bundled with related software and sold with a shrinkwrap license prohibiting all reverse engineering.[120] Baystate was a competitor in the CAD industry.[121] Shortly after Bowers began selling his bundle, Baystate acquired copies of it and within months had introduced its own substantially revised product incorporating features of Bowers's improvements.[122] Bowers brought breach of contract claims against Baystate for violating the terms of the shrinkwrap agreement.[123]

On appeal, the Federal Circuit applied First Circuit law in deciding the contract issue because that issue was not unique to its own jurisdiction.[124] Baystate argued that the shrinkwrap reverse engineering prohibition was preempted by the Copyright Act, but the Federal Circuit disagreed and affirmed the district court's judgment.[125] The Federal Circuit noted its respect for freedom of contract before squarely addressing the preemption issue.[126] It explained that § 301(a) of the Copyright Act (requiring that all exclusive rights within the general scope of copyright were governed exclusively by the Act) did not require preemption so long as a state cause of action required some non-illusory extra element.[127] It relied in part on *ProCD*'s holding that shrinkwrap licenses were not preempted by federal copyright law, noting the Seventh Circuit's observation that a copyright is a right against the world while contracts bind only their parties.[128] And it distinguished *Vault* on the basis that it was limited to a state law prohibiting copying of a computer program rather than "private contractual agreements supported by mutual assent and consideration."[129] It therefore held that Bowers's license terms forbidding reverse engineering were enforceable, and that Baystate was liable for breach of contract.[130]

119. Bowers v. Baystate Techs., Inc., 320 F.3d 1317, 1323 (Fed. Cir. 2003).
120. *Id.* at 1320–22.
121. *See id.* at 1322.
122. *Id.*
123. *See id.*
124. *Id.* at 1322–23.
125. *Id.* at 1323.
126. *Id.* at 1323–24.
127. *Id.* at 1324 (citing Data Gen. Corp. v. Grumman Sys. Support Corp., 36 F.3d 1147, 1164 (1st Cir. 1994)).
128. *Id.* (citing ProCD, Inc. v. Zeidenberg, 86 F.3d 1447, 1454 (7th Cir. 1996)).
129. *Id.* at 1325.
130. *Id.* at 1326.

Judge Dyk dissented on the preemption issue, noting that the majority's holding conflicted with the Fifth Circuit's holding in *Vault* and expressing concern that this logic threatened copyright policies generally.[131] He disagreed that shrinkwrap agreements were freely negotiated, likening them instead to contracts of adhesion where purchasers had no other choice but to enter into the contract or avoid buying the product altogether.[132] And he disagreed with the majority's reasoning that there was any difference between preemption of a state statute, as in *Vault*, and general common law contract principles.[133]

The Federal Circuit's majority opinion in *Bowers* has proved controversial.[134] Nonetheless, other circuits have followed it as best illustrated by *Davidson & Assocs. v. Jung*.[135] In *Davidson*, the defendants violated the terms of a shrinkwrap agreement when they reverse engineered the plaintiff's software to construct an emulator whereby plaintiff's games could be played for free.[136] The defendants relied on *Vault* in arguing that the Copyright Act preempted the plaintiff's state law breach-of-contract claims.[137] But the Eighth Circuit sided with the plaintiffs, finding the contract provision valid and enforceable.[138] As in *Bowers*, the court distinguished *Vault* on the basis that it addressed a particular state statute rather than a state law contract issue.[139] And it expressly relied on *Bowers* in determining that the defendants could contractually forego their reverse engineering rights under the Copyright Act.[140] It concluded that the defendants had indeed relinquished their reverse engineering rights when they agreed to the plaintiff's end-user license agreement (EULA), and that the defendants therefore breached the agreement.[141]

The facts underlying the reverse engineering in *Davidson* were strikingly similar to those of an earlier copyright case where the court reached the opposite result. In *Sony Computer Entertainment, Inc. v. Connectix Corp.*, Sony sued after Connectix reverse engineered Sony's

131. *Id.* at 1335 (Dyk, J., concurring in part and dissenting in part).
132. *Id.* at 1337.
133. *Id.*
134. *See, e.g.*, Robert W. Gomulkiewicz, *Fostering the Business of Innovation: The Untold Story of* Bowers v. Baystate Technologies, 7 WASH. J.L. TECH. & ARTS 445, 446–47 (2012).
135. 422 F.3d 630 (8th Cir. 2005).
136. *Id.* at 636.
137. *Id.* at 638.
138. *Id.* at 639.
139. *Id.*
140. *Id.*
141. *Id.*

PlayStation software to emulate the PlayStation on other platforms.[142] The Ninth Circuit held that the fair use exception to the Copyright Act protected the defendant's reverse engineering.[143] The key distinction between the cases, however, is that *Sony* did not involve a shrinkwrap agreement. The Eighth Circuit reached its conclusion in *Davidson* by applying contract principles, holding that the defendants breached the EULA when they reverse engineered the plaintiff's product.

Bowers and *Davidson* represent an encouraging trend for software companies seeking to protect their intellectual property through shrinkwrap agreements prohibiting reverse engineering. First the Federal Circuit, and then the Eighth Circuit, upheld broad prohibitions on reverse engineering, finding no preemption by federal copyright law. Yet these cases represent the positions of only two circuits—*Vault* is still the law in the Fifth Circuit, despite being nearly thirty years old. Furthermore, these cases applied copyright law rather than trade secret law, so exactly how these Circuits might handle trade secret protection is unclear. The following sections explore how software companies might best approach trade secret protection to maximize their chances of having contractual reverse engineering restrictions upheld by the courts.

B. Application to Trade Secret Law

In considering how contract principles apply to trade secrets, it is first important to note that contracts inherently play a more important role in trade secrets than they do in copyright or patent law. The UTSA, for instance, defines "improper means" as including "breach or inducement of a breach of a duty to maintain secrecy."[144] The DTSA, which is closely based on the UTSA, contains identical language.[145] Therefore when one party knows that it is receiving secret information from another, and expressly agrees to keep it secret by not performing reverse engineering, such contractual terms are generally enforceable.[146] Disclosure of trade secrets in violation of non-compete clauses and other similar contractual provisions between employees and their former employers comprises the vast majority of trade secret litigation.[147] At

142. Sony Comput. Entm't, Inc. v. Connectix Corp., 203 F.3d 596, 598 (9th Cir. 2000) (by reverse engineering Sony's software, Connectix made it possible for gamers to play Sony's PlayStation games on their own computers).

143. *Id.* at 609.

144. UNIF. TRADE SECRETS ACT WITH 1985 AMENDMENTS § 1(1) (UNIF. LAW COMM'N 1985).

145. 18 U.S.C.A. § 1839(6)(A) (West Supp. 2016).

146. *See* MILGRIM, *supra* note 4, §1.05(5)(III)(C)(1).

147. *See* David S. Almeling et al., *A Statistical Analysis of Trade Secret Litigation in State*

first blush, then, contractual provisions further limiting trade secret disclosure, such as through reverse engineering, seem to fit well in trade secret law, perhaps even better than with copyrights.

Can it be this simple? Can software companies simply argue that preemption by the DTSA is not an issue, and count on the courts to extend *Bowers* and *Davidson* to trade secret protection? Unfortunately for software owners, probably not. Breach of contractual agreements to maintain secrecy is part of the definition of trade secret misappropriation, but trade secret law has always taken a permissive approach toward reverse engineering.[148] Banning reverse engineering outright, even through contract, would therefore run counter to many decades of precedent in this area.

Shrinkwrap agreements also present unique problems. First, some courts might follow Judge Dyk and refuse to enforce them on the basis that they are contracts of adhesion. Fortunately for software owners, Judge Dyk's opinion appears to be the minority view. The majorities in *Bowers* and *Davidson* had no issue upholding a shrinkwrap agreement, and even the Fifth Circuit in *Vault* struck down the agreement on grounds of preemption rather than adhesion. Some members of the U.S. Supreme Court have also indicated that such contracts are enforceable so long as they are reasonable.[149]

The more significant problem with shrinkwrap agreements, and the one considered here, is that wide distribution of a software program with a shrinkwrap agreement prohibiting reverse engineering may be challenged as an improper attempt to extend a monopoly right that is only granted by patents, not trade secrets. Patents are more closely related to trade secrets than copyrights because patents and trade secrets cover very similar subject matter. Copyrights are limited to artistic expression; they cannot cover ideas or anything having utility.[150] Patents, by contrast, can cover "anything under the sun made by man."[151]

Courts, 46 GONZ. L. REV. 57, 59–60 (2011) (noting that the "vast majority" of alleged misappropriators were employees or business partners).

148. *See, e.g.*, Kewanee Oil Co. v. Bicron Corp., 416 U.S. 470, 476 (1974) ("A trade secret law, however, does not offer protection against discovery by fair and honest means, such as by independent invention, accidental disclosure, or by so-called reverse engineering . . . .").

149. *See, e.g.*, Carnival Cruise Lines, Inc. v. Shute, 499 U.S. 585, 600 (1991) (Stevens & Marshall, JJ., dissenting).

150. 17 U.S.C. § 102(b) (2012) ("In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work.").

151. Diamond v. Chakrabarty, 447 U.S. 303, 309 (1980) (citing S. REP. NO. 82-1979, at 5 (1952) and H.R. REP. NO. 82-1923, at 6 (1952)).

According to the DTSA, trade secrets can cover "all forms and types of financial, business, scientific, technical, economic, or engineering information."[152] There is therefore substantial overlap between patent and trade secret protections, and the more relevant question in assessing the validity of shrinkwrap agreements in the trade secret context may be preemption by federal patent law rather than copyright law. As discussed below, the best solution to all of these problems may be a narrow shrinkwrap provision that prohibits reverse engineering only for certain purposes. To get there, however, it is necessary to first consider the merits of a complete contractual ban on reverse engineering to discern trade secrets in software.

1. Total Contractual Ban on Reverse Engineering

The first question for software companies to consider is whether a shrinkwrap license prohibiting reverse engineering of trade secret-protected code for any purpose whatsoever would survive judicial scrutiny. As discussed above, the courts have thus far only analyzed this question in the context of federal copyright law, not trade secret law.[153] *Vault*, *Bowers*, and *Davidson* were copyright cases, and there is no indication that the plaintiffs ever argued that their software was a trade secret or that the defendants had misappropriated it.

The first issue with a total contractual ban on reverse engineering to discern trade secrets is whether it might be preempted by federal trade secret law, just as *Vault*, *Bowers*, and *Davidson* examined preemption of contractual bans by federal copyright law. Federal trade secret law is now governed by the DTSA, which is closely modeled on the UTSA. Unlike the UTSA, however, the DTSA expressly excludes reverse engineering from "improper means," meaning that reverse engineering is a legitimate means of discerning trade secrets under federal law.[154]

As for preemption, the DTSA states that "this chapter [18 U.S.C. §§ 1831 *et seq.*] shall not be construed *to preempt or displace* any other remedies, whether civil or criminal, provided by United States Federal, State, commonwealth, possession, or territory law for the misappropriation of a trade secret."[155] This passage has not yet been tested in the courts, but it appears from the plain meaning that the non-preemption condition is limited to state trade secret law and therefore

152. 18 U.S.C.A. § 1839(3) (West Supp. 2016).
153. *See supra* Section III.A.
154. 18 U.S.C.A. § 1839(6)(B) (West Supp. 2016).
155. 18 U.S.C.A. § 1838 (West Supp. 2016) (emphasis added).

would not extend to contractual bans. But it expressly states that it does not affect state law trade secret protections, showing that it is meant to coexist with state protections and have a much narrower scope than, for instance, federal patent law. More broadly, a software company trying to protect its trade secrets through a contractual ban on reverse engineering could still rely on *ProCD* to argue that trade secret rights, like rights under the Copyright Act, are rights against the world.[156] Trade secret rights, after all, "restrict the options of persons who are strangers to the author."[157] Therefore the additional contractual ban could be justified under the logic of *ProCD*.

Although software companies seeking to protect their trade secrets through contractual bans on reverse engineering have strong arguments that there is no preemption by federal trade secret law, preemption by federal patent law is altogether different and much more challenging. Two major Supreme Court decisions examining preemption by patent law, *Bonito Boats, Inc. v. Thunder Craft Boats, Inc.* and *Kewanee Oil Co. v. Bicron Corp.*, illustrate the likely challenges such a ban would face.[158]

In *Bonito Boats*, the Court struck down a Florida statute prohibiting the use of a particular molding process used to duplicate unpatented boat hulls.[159] In doing so, the Court noted that the Patent Clause of the United States Constitution carefully balances monopoly rights that stifle competition against public disclosure that allows for a competitive economy.[160] The Court further noted the importance of committing an invention to the public after its patent expires, faulting the statute for protecting unpatented designs in perpetuity.[161] The Court rejected the respondent's argument that the Florida law at issue was needed to protect against "unfair competition."[162] The Court held instead that the law was preempted by federal patent law because it upset patent law's balance of rights by providing a monopoly right in a substantive idea with no return for the public.[163]

156. ProCD, Inc. v. Zeidenberg, 86 F.3d 1447, 1454 (7th Cir. 1996).
157. *Id.*
158. Bonito Boats, Inc. v. Thunder Craft Boats, Inc., 489 U.S. 141 (1989); Kewanee Oil Co. v. Bicron Corp., 416 U.S. 470 (1974).
159. *Bonito Boats*, 489 U.S. at 168.
160. *Id.* at 146.
161. *Id.* at 159.
162. *Id.* at 157–58.
163. *Id.* at 159–60.

Despite the Court's emphasis on the delicate balance of rights established by patent law, software companies can likely distinguish a total contractual ban on reverse engineering from *Bonito Boats*. To do so, they could point to the Court's emphasis on the importance of *disclosure* to the public in exchange for the monopolistic right to exclude others from using an invention.[164] The Court described this exchange in stating that "[a] state law that substantially interferes with the enjoyment of an unpatented . . . conception which has been freely disclosed by its author to the public at large impermissibly contravenes the ultimate goal of *public disclosure* and use which is *the centerpiece of federal patent policy*."[165] Patent law does indeed have substantial public disclosure requirements. It requires that a patent specification contain a written description clearly disclosing the patentable aspects of the invention, and that all patent claims be sufficiently enabled by the patent specification such that a person of ordinary skill in the relevant field can make and use the invention.[166] But trade secret law points the opposite way, requiring privacy rather than public disclosure. Specifically, the UTSA requires that trade secret owners take reasonable measures to maintain secrecy,[167] and courts have held that shipping software in object code format meets that requirement.[168] Software companies can therefore distinguish their trade secrets from the publicly displayed boat hulls in *Bonito Boats* by pointing out that trade secrets are not in the "public domain." Rather, trade secrets are necessarily subject to efforts to maintain secrecy from the public, specifically by circulating the software as indecipherable object code. Extending protections for trade secrets, for instance by using shrinkwrap agreements to ban reverse engineering, therefore does not upset the delicate balance of patent law in the way that the Court feared it would.

Other parts of *Bonito Boats* are more difficult to distinguish, however, particularly when contemplating a shrinkwrap provision imposing a *total* ban on reverse engineering. Although *Bonito Boats* did

164. *Id.* at 150–151 ("The federal patent system thus embodies a carefully crafted bargain for encouraging the creation and *disclosure* of new, useful, and nonobvious advances in technology and design in return for the exclusive right to practice the invention for a period of years. . . . [T]he ultimate goal of the patent system is to bring new designs and technologies into the public domain through *disclosure*." (emphasis added)).

165. *Id.* at 156–57 (emphasis added).

166. 35 U.S.C. § 112(a) (2012).

167. UNIF. TRADE SECRETS ACT WITH 1985 AMENDMENTS § 1(4)(ii) (UNIF. LAW COMM'N 1985) (defining a "trade secret" in part as "the subject of efforts that are reasonable under the circumstances to maintain its secrecy").

168. *E.g.*, Q-Co Indus. v. Hoffman, 625 F. Supp. 608, 617–18 (S.D.N.Y. 1985).

not directly concern trade secrets, the Court addressed the close relationship between trade secrets and reverse engineering in its reasoning. It faulted the Florida law for prohibiting the public from engaging in reverse engineering of products, noting that such prohibitions had never been part of state trade secret law.[169] And the Court specifically cited *Kewanee* as standing for the proposition that trade secret law does not protect against discovery by reverse engineering.[170] Some authors therefore cite *Bonito Boats* as a reason for prohibiting bans on reverse engineering in relation to trade secrets.[171]

Kewanee provides a similar perspective on blanket contractual bans on reverse engineering of trade secrets, and probably stands as the greatest impediment to such a ban. There the Court held that federal patent law did not preempt Ohio trade secret law because trade secret protection is so much weaker than patent protection.[172] As an example of this relative weakness, the Court explained that reverse engineering was a proper means of obtaining trade secrets, but was not a justification for patent infringement.[173] Trade secret law, the Court went on, therefore functioned more like a "sieve" compared with patent law's "barrier."[174] Banning reverse engineering outright would seem to convert the sieve into a barrier and might lead courts to conclude that such bans, even though rooted in contract, are preempted by federal patent law.

While the Supreme Court has not addressed trade secrets and preemption by patent law in any detail since *Bonito Boats* and *Kewanee*, other courts have noted that those decisions make it unlikely that a total contractual ban on reverse engineering to discern trade secrets would survive. In *DVD Copy Control v. Bunner*, for instance, the California Supreme Court relied on *Bonito Boats* in dicta when it found that banning reverse engineering of trade secrets through contract (in this case, by using a form contract to redefine "improper means" of determining trade secrets to include reverse engineering) would probably be preempted by federal patent law.[175] Software companies seeking to protect their trade secrets are therefore unlikely to succeed in completely

169. *Bonito Boats*, 489 U.S. at 160.

170. *Id.* (citing Kewanee Oil Co. v. Bicron Corp., 416 U.S. 470, 476 (1974)).

171. *See, e.g.*, Pamela Samuelson, *Reverse Engineering Under Siege*, 45 COMM. OF THE ACM, no. 45, 2002, at 15, 16–17 (noting the importance of *Bonito Boats* to any reverse engineering public policy analysis).

172. *Kewanee*, 416 U.S. at 489–90.

173. *Id.* at 490.

174. *Id.*

175. DVD Copy Control Ass'n v. Bunner, 75 P.3d 1, 28 n.5 (Cal. 2003) (citing *Bonito Boats*, 489 U.S. at 155).

banning reverse engineering of their products through shrinkwrap agreements.

2. Limited Contractual Ban on Reverse Engineering

Software companies need not advocate for a blanket ban on reverse engineering, however. They could instead include a term in a shrinkwrap agreement that bans reverse engineering of the software for specific limited purposes such as direct economic competition. Such a limited ban would be easier to distinguish from both *Kewanee* and *Bonito Boats*, and there are statutory analogues in copyright law to which software companies could point as well.

a. Case Law

The greatest case law challenges even to a limited contractual ban on reverse engineering are still *Bonito Boats* and *Kewanee*. Those two cases are closely intertwined because *Bonito Boats* relied extensively on *Kewanee*, particularly when addressing the prohibitions on reverse engineering that the Supreme Court ultimately found improper.[176] *Kewanee* appears to be the greatest obstacle to a total contractual ban on reverse engineering because it would strengthen the protections of trade secret law too much. But a far more limited and narrowly targeted contractual ban on reverse engineering would strengthen trade secret law only slightly. If the shrinkwrap provision prohibited reverse engineering only for direct economic competition, the public would still have the benefit of reverse engineering for adaptability and other purposes.

In *Kewanee*, the Court noted that trade secret law's permissive approach toward reverse engineering distinguished it from patent law, but the Court did not address the degree to which reverse engineering might be restricted. In *Bonito Boats*, the Court appeared to go beyond *Kewanee* by laying out in more detail what sorts of bans on reverse engineering would be preempted by patent law. Specifically, the Court suggested that even limited bans on reverse engineering might be problematic.[177] According to the Court, that the Florida statute did not remove *all* means of reproduction was not enough to save it.[178] The Court described the law as prohibiting the public from engaging in "a form of reverse engineering of a product in the public domain," and said

176. *Bonito Boats*, 489 U.S. at 160.
177. *See id.* at 160.
178. *Id.*

that this was a right of a patent holder, but was not part of trade secret law.[179] The Court went on to address benefits of reverse engineering, specifically noting that it could lead to "significant advances in technology" by incentivizing the inventor to continue to make improvements and receive a patent.[180]

Even here, software companies seeking to protect their trade secrets from reverse engineering should find it easier to distinguish a limited ban than a broad one. As with the total ban, these companies could point out that their underlying algorithms are not in the "public domain" in the conventional sense, because they are circulated in an obfuscated form. This measure is enough to qualify for trade secret protection, so it can hardly be said to be public in the same way that the boat hulls in *Bonito Boats* were placed in plain view. Second, these companies can point out that prohibiting reverse engineering for a narrow purpose, such as direct economic competition, does not frustrate innovation because other companies would still be free to reverse engineer the software to improve upon it. This still doesn't offer software companies the level of protection they might have had in these algorithms before *Alice*, but it offers additional protection nonetheless. Furthermore, the impact of *Alice* is ultimately the key point—in a post-*Alice* world, these contractual trade secret protections still do not rise to the level of patent protection and therefore are not preempted by federal patent law.

Finally, any defendant accused of violating such a shrinkwrap provision to discern trade secrets would almost certainly rely on *Vault*. It is the primary case from the circuit courts that invalidated shrinkwrap provisions prohibiting reverse engineering because they were preempted by federal intellectual property law, and it appears to still be good law in the Fifth Circuit. However, the Federal Circuit in *Bowers* may have interpreted *Vault* too broadly, and software companies seeking to protect their trade secrets could urge a narrower interpretation.

Vault is often characterized as holding that a state law prohibiting all copying of a computer program is preempted by the Copyright Act.[181] But the statutory language at issue in *Vault* was actually much narrower. The LSLEA specifically prohibited "*adaptation* by reverse engineering, decompilation or disassembly," that is, modifying the software to serve

179. *Id.*

180. *Id.* at 160–61.

181. *See, e.g.*, Bowers v. Baystate Techs., Inc., 320 F.3d 1317, 1338 (Fed. Cir. 2003) (Dyk, J., concurring in part and dissenting in part) ("I conclude that *Vault* states the correct rule; that state law authorizing shrinkwrap licenses that prohibit reverse engineering is preempted . . . .").

some independent purpose.[182] But the LSLEA did not address reverse engineering in any other context.[183] The Fifth Circuit focused heavily on this "adaptation" language in its preemption finding. For instance, it pointed to § 117 of the Copyright Act which it noted "permits an owner of a computer program to make an *adaptation* of that program . . . ."[184] It emphasized the adaptation limitation yet again in restating its holding when it said "[t]he provision in Louisiana's License Act, which permits a software producer to prohibit the *adaptation* of its licensed computer program by decompilation or disassembly, conflicts with the rights of computer program owners under § 117 and clearly 'touches upon an area' of federal copyright law."[185]

Had the LSLEA allowed shrinkwrap agreements to prohibit reverse engineering of a computer program for the narrower purpose of creating a directly competing version, it is less clear how the Fifth Circuit would have ruled. The Copyright Act does not expressly address the creation of directly competing products in any of its fair use provisions. Furthermore, prohibiting software owners from barring direct copying runs counter to the control that the Copyright Act provides authors and artists to determine how their work is reproduced and disseminated. The Fifth Circuit might well have found that such a version of the LSLEA statute—and therefore the license—was valid.

b. Statute

Moving beyond case law, at least one statutory analog suggests that a limited ban is viable. The Digital Millenium Copyright Act (DMCA), passed into law in 1998, amended U.S.C. Title 17 to add restrictions on circumvention of access control measures.[186] The DMCA effectively barred reverse engineering for the purpose of circumventing access controls by expressly permitting reverse engineering only if it was used for interoperability purposes traditionally covered by fair use.[187] With the DMCA, Congress responded to the concerns of game developers and other software companies that their competitors, and in many cases their customers, were circumventing access controls to create free versions of

182. Vault Corp. v. Quaid Software Ltd., 847 F.2d 255, 269 (5th Cir. 1988) (emphasis added).
183. *Id.*
184. *Id.* at 270 (emphasis added).
185. *Id.* (emphasis added).
186. 17 U.S.C. § 1201(a)(1)(A) (2012).
187. *See id.* § 1201(f).

protected software.[188] The DMCA's restrictions on reverse engineering were directed at stopping such circumvention.[189] Although it included fair use exceptions similar to those in the existing Copyright Act, such as for interoperability, the existence of the DMCA demonstrates that the federal government is concerned about certain kinds of competition arising from reverse engineering.

The competition that gave rise to the DMCA was similar to the problem software companies likely face from competitors looking to steal their trade secrets, in that it involved direct economic competition and lost profits. The DMCA suggests that the government is at least somewhat sympathetic to this problem and willing to restrict certain kinds of reverse engineering to address it. Although no comparable statute exists for trade secrets, software companies seeking to protect their secrets through limited contractual bans on reverse engineering for direct economic competition could point to the DMCA as an example of a successful approach that balances incentives for innovation against complete monopolies.

But just how far might a contractual ban on reverse engineering reasonably go? There is no fair use provision in trade secret law comparable to the express fair use provisions of the Copyright Act and its amendments, so it is difficult to know just where the line would be drawn even if a court followed *Vault* rather than *Bowers*. One software-related argument for fair use in the copyright context is that copyright protects only expression and not ideas. Banning reverse engineering of copyrighted software therefore improperly protects unprotectable ideas along with protectable expression.[190] Trade secrets, unlike copyrights, *can* protect ideas—the UTSA and DTSA include devices, methods, techniques, and processes as protectable subject matter, all of which embody ideas.[191] There is therefore less need for fair use in trade secrets. This may weigh toward upholding broader bans on reverse engineering in the trade secret context, and is at least another way for software companies to distinguish the copyright analysis in *Vault*.

In view of these arguments and the various cases distinguished, limited contractual bans on reverse engineering appear likely to pass

188. Donna L. Lee, Comment, *Reverse Engineering of Computer Programs Under the DMCA: Recognizing a "Fair Access" Defense*, 10 MARQ. INTELL. PROP. L. REV. 537, 550 (2006).

189. *Id.* at 552–53.

190. *See* MILGRIM, *supra* note 4, § 1.05(5)(II)(2) (explaining how prohibiting decompilation might allow copyright owners to monopolize ideas and expression).

191. 18 U.S.C.A. § 1839(3) (West Supp. 2016); UNIF. TRADE SECRETS ACT WITH 1985 AMENDMENTS § 1(4) (UNIF. LAW COMM'N 1985).

judicial scrutiny. Software companies should be cautious, however, not to extend their bans too broadly.

IV. CONCLUSION

The Supreme Court's decision in *Alice Corp. v. CLS Bank* dealt a significant blow to software companies looking to protect their intellectual property, but trade secrets appear to offer a way forward. As these companies evaluate how to protect their software algorithms as trade secrets, they should consider how best to prevent competitors from reverse engineering their products. Trade secret law does not itself protect against reverse engineering, but contract law can. In applying contract principles, software companies would do best to avoid total bans on reverse engineering. But selective bans on reverse engineering for certain purposes such as development of directly competing products appear much more likely to survive judicial scrutiny. They are consistent with recent Circuit decisions in copyright law, do not directly contravene any statutes, and appear to avoid the most significant public policy concerns regarding reverse engineering.

A more limited ban does not create a monopoly right on par with the near-total exclusive rights that patent protection confers, and therefore should not conflict with *Bonito Boats* while respecting the more tenuous nature of trade secret protections versus patent rights described in *Kewanee*. Such a limited ban therefore seems to strike a balance between good policy and conformance with case law and statute. Software companies looking to protect their investments in the wake of *Alice* might do well to start there.