Berkeley Technology Law Journal

Volume 13 | Issue 1 Article 26

January 1998

Bernstein v. United States Department of State

Patrick Ian Ross

BERNSTEIN V. UNITED STATES DEPARTMENT OF STATE

By Patrick Ian Ross

Is computer source code a constitutionally protected form of "speech"? Do government controls on the export of electronic source code violate a programmer's constitutional rights? The court in *Bernstein v. United States Department of State*[1] decided the answer to both questions is "yes." The court reasoned that source code is a language and therefore, by definition, speech. It bolstered this claim with analogies to recipes, music, and copyright law.

If the court is right, and electronic source code is speech, then the government's power to regulate its distribution will be greatly restricted. Distributing source code—that is, "speaking"—becomes a constitutionally protected activity. Further, government licensing requirements for the export of encryption software, like those recently declared unconstitutional,[2] are prior restraints with a "'heavy presumption' against [their] constitutional validity."[3]

Although it may have reached the right judgment, the *Bernstein I* court decided too quickly that source code is by definition speech. Such a holding is too sweeping in scope. What matters most in deciding whether source code is speech is (1) the form in which computer code is distributed, and (2) to whom it is addressed. Source code is not automatically a "language" that should be afforded First Amendment protection because it is often distributed in executable form and used simply as a machine, and not used as communication between humans.

I. BACKGROUND

Daniel Bernstein, while a graduate student and teaching assistant at the University of California at Berkeley, wrote a strong encryption algorithm

1. 922 F. Supp. 1426 (N.D. Cal. 1996) [hereinafter *Bernstein I*].
2. *See* Bernstein v. United States Dep't of State, 945 F. Supp. 1279 (N.D. Cal. 1996) [hereinafter *Bernstein II*]; Bernstein v. United States Dep't of State, 974 F. Supp. 1288 (N.D. Cal. 1997) [hereinafter *Bernstein III*].
3. Organization for a Better Austin v. Keefe, 402 U.S. 415, 419 (1971) (quoting Carroll v. President and Comm'rs of Princess Anne, 393 U.S. 175, 181 (1968)); *see also* FW/PBS, Inc. v. Dallas, 493 U.S. 215 (1990).

he named "Snuffle."[4] He wrote it in the high-level computer language "C." Bernstein engineered Snuffle to be capable of both encrypting and decrypting data.[5]

Encryption is an application of the science of cryptography. "Encryption basically involves running a readable message known as 'plaintext' through a computer program that translates the message according to an equation or algorithm into unreadable 'ciphertext;'"[6] decryption reverses the process. Cryptography has several uses. It ensures data integrity (prevents changing or tampering with a message), authenticates users (stamps a message with a reliable digital signature), facilitates nonrepudiation (links a specific message with a specific sender), and maintains confidentiality (makes a message unreadable to all but the intended recipient).[7] The technology has obvious uses in an electronic age, for it allows one to make reliable and secure digital transactions.

Unreadable messages, however, threaten the government's surveillance power. Encryption technology enables foreigners to communicate securely by sending messages safe from detection by federal agencies responsible for monitoring such communications. As a result, the United States has for many years regulated the export of encryption software for reasons of national security. The United States government requires anyone to obtain an export license if he or she wants to distribute strong encryption software in a way that can reach foreigners.[8] Posting software on the Internet is an example. These government regulations stop most distribution of strong encryption software because most distribution channels can be accessed by foreigners. The government agencies responsible for regulating the export of encryption were given absolute discretion to decide whether to grant export licenses.[9] They had no deadlines for making

4. *See Bernstein I*, 922 F. Supp. at 1429.
5. *See id.*
6. *Bernstein II*, 945 F. Supp. at 1282.
7. *See id.*
8. When this litigation began, the Department of State regulated the export of encryption software by classifying it as a munition. *See* International Traffic in Arms Regulations (ITAR), 22 C.F.R. §§ 120-30 (1994). Just before the court invalidated these regulations in *Bernstein II*, President Clinton shifted encryption export controls to the Department of Commerce. *See* Exec. Order No. 13,026, 21 Fed. Reg. 58,767 (1996) (order authorizing shift of export controls from State to Commerce); 61 Fed. Reg. 68,572 (1996), (to be codified at 15 C.F.R. pts. 730-774) (regulation implementing executive order). The new Department of Commerce regulations were struck down in *Bernstein III*. *See* 974 F. Supp. at 1310.
9. *See Bernstein II*, 945 F. Supp. at 1289.

a decision, no guidelines were given (so decisions could be arbitrary), and the governing statute explicitly precluded judicial review.[10]

Bernstein wanted to post the Snuffle algorithm on the Internet so his students could access and download it.[11] Accordingly, he submitted a commodity jurisdiction request to the Department of State to ascertain whether Snuffle was subject to government regulation.[12] The Department of State replied that Snuffle was subject to regulation, and denied Bernstein permission to distribute his software.[13] After many months of "copious and contentious correspondence,"[14] Bernstein filed suit, claiming the government's licensing scheme violated his First Amendment rights.

In *Bernstein I,* the District Court for the Northern District of California ruled on the Department of State's motion to dismiss for lack of justiciability. In the course of this opinion, the court held that source code is speech for First Amendment purposes.[15] In *Bernstein II & III,* involving cross-motions for summary judgment, the court held that the government licensing schemes, which it called "a paradigm of standardless discretion," constituted illegal prior restraint on speech.[16] The interesting holding is that of *Bernstein I*—that source code is speech.[17]

10. *See id.* at 1289-90.
11. *See id.* at 1296.
12. *See Bernstein I,* 922 F. Supp. at 1430.
13. *See id.* Bernstein also submitted commodity jurisdiction requests for three accompanying texts, written in English, which described how the Snuffle system works and how it is used. *See id.* The Department of State originally claimed to regulate not only Bernstein's source code but also his English papers, and it denied him permission to distribute anything. *See id.* at 1433-34. Approximately two years later, after Bernstein filed suit but before trial began, the Department of State disavowed this decision. *See id.* The court noted "[i]t is disquieting that an item defendants now contend could not be subject to regulation was apparently categorized as a defense article and subject to licensing for nearly two years, and was only reclassified after plaintiff initiated this action." *Id.* at 1434.
14. *Id.* at 1430.
15. *See id.* at 1436.
16. *Bernstein II,* 945 F. Supp. at 1289; *see Bernstein III,* 974 F. Supp. at 1308.
17. Notice that the issue in *Bernstein* is not whether the government may keep someone from "exporting" an encrypted *message.* The issue is *not* whether one can speak confidentially. The use of encryption software is not addressed by this case, only its export.

II. THE COURT'S REASONING

The court decided very quickly that source code is speech. It dismissed the government's conduct analysis,[18] stating that "[i]t would be convoluted indeed to characterize Snuffle as conduct in order to determine how expressive it is when, at least formally, it appears to be speech."[19] The court relied on the following language from *Yniguez v. Arizonans for Official English*[20] in making its decision:

> Of course, speech in any language consists of the "expressive conduct" of vibrating one's vocal chords, moving one's mouth and thereby making sounds, or of putting pen to paper, or hand to keyboard. Yet the fact that such "conduct" is shaped by language—that is, a sophisticated and complex system of understood meanings—is what makes it speech. Language is by definition speech, and the regulation of any language is the regulation of speech.[21]

The court applied the reasoning of *Yniguez* to source code in this crucial paragraph: "[T]he particular language one chooses [does not] change the nature of language for First Amendment purposes. This court can find no meaningful difference between computer language, particularly high-level languages ... and German or French. All participate in a complex system of understood meanings within specific communities."[22]

The court presumably reached this conclusion because source code contains "[t]he expression of ideas, commands [and] objectives,"[23] and can be read and understood by computer programmers, although it does not explain its reasoning in any detail. The holding is quite sweeping, for the court would extend the reasoning even to object code, a form of software usually read by computers, but very rarely by humans:

> Even object code, which directly instructs the computer, operates as a "language." When the source code is converted into the object code "language," the object program still contains the text

18. It should be noted that even conduct may be protected under the First Amendment if there is an intent to convey a particular message and the likelihood is great that the message will be understood. *See, e.g.*, Texas v. Johnson, 491 U.S. 397 (1987) (evaluating the communicative aspects of flag burning).

19. *Bernstein I*, 922 F. Supp. at 1435.

20. 69 F.3d 920 (9th Cir. 1995) (en banc), *vacated as moot*, __ U.S. __, 117 S. Ct. 1055 (1997).

21. *Bernstein I*, 922 F. Supp. at 1435 (quoting *Yniguez*, 69 F.3d at 934-35).

22. *Bernstein I*, 922 F. Supp. at 1435.

23. *Id.*

> of the source program. The expression of ideas, commands, objectives and other contents are merely translated into machine-readable code.[24]

The court defended its claim that computer code is language and language is speech with analogies to recipes, music, and copyright law. These analogies were meant to show that the functionality of code does not render it non-speech. First, the court pointed out that "[i]nstructions, do-it-yourself manuals, recipes, even technical information about hydrogen bomb construction"[25] are functional and also speech. Therefore, even if Snuffle "is essentially functional, that does not remove it from the realm of speech."[26]

The court strengthened the argument by analogizing source code to the music scroll of a player piano. "The music inscribed in code on the roll of a player piano is no less protected [by the First Amendment] for being wholly functional. Like source code converted to object code, it 'communicates' to and directs the instrument itself, rather than the musician, to produce the music."[27] Both the recipe and music analogies support the proposition that a "language" that actually allows one to do something, like "play music or make lasagna,"[28] or indeed, encrypt data, is nonetheless speech.

The court made a final analogy to copyright law. The court argued that software, whether defined as source code or object code, is protected by copyright. Copyright protects the "expression" of a computer program. It follows that "[t]he 'expression' of an idea ... connotes the 'speaking' of an idea."[29] The court recognized the danger in this analogy, but concluded that it still provided support for its decision. "While copyright and First Amendment law are by no means coextensive, and the analogy between them should not be stretched too far, copyright law does lend support to the conclusion that source code is a means of original expression."[30]

24. *Id.*
25. *Id.*
26. *Id.*
27. *Id.* Music is speech protected by the First Amendment. *See* Ward v. Rock Against Racism, 491 U.S. 781, 790 (1989).
28. *Bernstein I,* 922 F. Supp. at 1436.
29. *Id.*
30. *Id.*

III. DISCUSSION

The court explicitly offered these three analogies to show only that source code's functionality does not render it non-speech. This limited claim is true. But the court's analogies do more than this—they also implicitly operate to persuade us that source code *is* speech. The analogies are all we have to rely on; nothing else is offered in support of the holding beyond the court's statement that "at least formally, [source code] appears to be speech."[31]

But under close examination, the court's analogies neither persuade that source code is speech nor provide any meaningful critical tools for deciding the issue. The court's analogies miss the underlying question whether code is or is not a form of human communication. This question must be addressed directly through a factual analysis of the form in which source code is distributed and to whom it is addressed.

A. Counteranalogy[32]

Part of the problem with the court's analysis is its exclusive reliance on analogies. Unfortunately, analogies can easily be fashioned to support either the argument that computer code is speech or that it is not. To illustrate the point, consider that encryption software is like a machine that places messages into locked envelopes. Readable messages are put into an unreadable, encoded (locked) form.[33] It is important to understand that computer software—including encryption software—is functionally indistinguishable from hardware. Software is like a sophisticated machine engineered to perform a specific task. As leading authorities on computer technology and law have written,

> [P]rograms are, in fact, machines (entities that bring about useful results, i.e. behavior) that have been constructed in a medium of text (source and object code). The engineering designs embodied in programs could as easily be implemented in hardware as

31. *Id.*

32. In this and the subsection that follows, the terms "source code," "electronic source code," and "software" are used semi-interchangeably. The meaning of "source code" and "object code" are consistently conflated. These concepts contain differences that matter a great deal—in fact, the First Amendment analysis partly turns on these differences. These issues will be addressed infra.

33. As the court wrote, "Encryption basically involves running a readable message known as 'plaintext' through a computer program that translates the message according to an equation or algorithm into unreadable 'ciphertext.'" *Bernstein II,* 974 F. Supp. at 1282.

> software, and the user would be unable to distinguish between the two.[34]

Encryption software is therefore something like a lock-making machine.

Locks are not speech, and neither are the machines that make them. Although there may be an expressive element in locking something up (keep out), this element does not turn such conduct into speech.[35] The government may regulate the export of lock-making machines if it so desires.

The same analysis can be applied to encryption. But the analogy does not decide the issue, because it focuses on the functional aspects of software and not the communicative aspects of it. Therefore the analogy assumes what it purports to prove, and begs the question whether source code is speech. If read with skepticism, the court's analogies also miss or beg the question. And with a few tweaks, they can easily be used to support the opposite argument.

B. Criticism of the Court's Three Analogies

First, the court draws an analogy between source code and an instruction manual or recipe, which are certainly protected speech. But notice that recipes are not machines that perform tasks. Recipes convey information from one person to another—information that tells another person how to engage in conduct, perhaps how to build a machine. Source code, on the other hand, is much more like the machine itself. Consider an imperfect analogy: an instruction manual about how to build a bomb does not actually build the bomb, whereas cryptographic source code generates object code that actually encrypts.[36] The court uses the recipe analogy to

34. Pamela Samuelson et al., *A Manifesto Concerning the Legal Protection of Computer Programs,* 94 COLUM. L. REV. 2308, 2316 (1994).

35. Notice that most pure conduct can be described as having expressive elements. To cite an extreme example, political murder has quite a strong expressive element, but this does not make it speech for First Amendment purposes.

36. The analogy is imperfect because what encryption produces is not a weapon but confidential speech. The First Amendment status of confidential speech has not been adjudicated, although the right to speak anonymously has been upheld, as has the right to speak a language others do not understand. *See* McIntyre v. Ohio Elections Comm'n, 514 U.S. 334 (1995); Talley v. California, 362 U.S. 60 (1960) (upholding the right to speak anonymously); *see also* Yniguez v. Arizonans for Official English, 69 F.3d 920 (9th Cir. 1995) (en banc), *vacated as moot,* __ U.S. __, 117 S. Ct. 1055 (1997) (upholding the right to speak Spanish in an English-speaking environment). It is an interesting question whether confidential speech enjoys First Amendment protection, but this question, and the incidentally related question whether a right to speak confidentially extends to the export of encryption software, are beyond the scope of this comment.

conceal rather than highlight this difference. But the sophisticated behavior of a computer program, behavior that produces sophisticated results, strains the analogy to recipes precisely at the juncture between communication and function. By ignoring the interplay between these two characteristics, the court's analogy begs the question it is meant to answer.

Next, the court draws an analogy between source code and the music scroll of a player piano. But this analogy does not necessarily help the court because one can spin the analogy a number of different ways. Consider that encrypting code is like a music scroll-making machine: both change a message (data or music) into an encoded form. Similarly, decrypting code is like a player piano: both change an encoded message into data or music. One might conclude from this analysis that just as the government may regulate the export of music scroll-making machines or player pianos, it may regulate the export of cryptographic code. To continue spinning the analogy, perhaps music scrolls are not like source code but like encrypted messages, because they are encoded music. Or perhaps music scrolls are more like *decrypted* messages because any old player piano will play the music (music scroll as ASCII file). But if music scrolls are like messages then the analogy is not useful, because the status of messages was not at issue in *Bernstein*. Finally, perhaps music scrolls are in fact more like computer instructions than messages. After all, the scroll directs the piano like a program directs a computer.

The conceptual problem here is that music scrolls are hybrid objects. They are both a program and a message. Insofar as they are a message, they are protected speech; but insofar as they are just programs, they are unprotected machines. Similar reasoning can be applied to source code. But unfortunately, the music scroll analogy does not provide any basis for deciding whether source code is more like speech or more like machinery.[37] It only illustrates the problem.

Finally, the court argues that source code's protection under copyright law lends support to the claim that source code is speech. But consider that the creativity protected by software copyrights is not the same as the speech protected in books. Instead, the "expressiveness" of software means the ingenuity expressed in a complex engineering design. "For all

37. It would be a more difficult case if an encryption program were inextricably bound up with the text of the encrypted message in the way program and message are bound together in a music scroll.

its parallels to literature and art, computer programming is at base engineering—it involves the construction of a machine."[38]

Furthermore, it has long been recognized that copyright is not a perfect doctrinal box for protecting software for exactly this reason.[39] Therefore we should hesitate before drawing too many conclusions from the fact that software is protected this way. The *Bernstein* case can be read as an example of the absurdity we get if we take the relationship between software and copyright too seriously. Consider that the holding in *Bernstein I,* if taken to its logical conclusion, may create real doctrinal imprecision. For example, as software patents proliferate,[40] granting software First Amendment status becomes quite conceptually awkward. Consider the following counterintuitive proposition: if software is held to be speech, and software can be patented, then certain kinds of speech can be patented. Although this statement contains no logical inconsistency, it certainly muddies First Amendment doctrine. Again, the court's analogy merely flags, rather than resolves, the issues raised in this case.

Where does all this analogy spinning leave us? No further than where we began. The court's reliance on analogies that can easily be construed in a contrary fashion weakens its arguments. The question the court should have focused on is whether the source code is essentially communicative or functional.

C. Is Source Code Speech or Engineering?

The *Bernstein* court's analogies point to but do not directly tackle the real questions. The First Amendment issue turns on whether electronic source code is more like a machine or a language: more functional or communicative. But to whom is source code addressed? And in what sense is source code functional? The answers to these questions are critical to deciding if source code deserves constitutional protection as speech. If source code boils down to communication between humans, then it deserves First Amendment protection. But if it is merely the engineering of a machine, it does not.

Unfortunately, the problem does not admit of one answer. The status of source code varies from case to case because code is recorded in many

38. ROBERT P. MERGES ET AL., INTELLECTUAL PROPERTY IN THE NEW TECHNOLOGICAL AGE 850 (1997).

39. For an extended discussion of the need to develop a new intellectual property right specifically for software, see generally Samuelson, *supra* note 34.

40. For an example of how close courts have come to sidestepping the rule against algorithm patents laid down in *Gottschalk v. Benson,* 409 U.S. 63 (1972), see *In re Alappat,* 33 F.3d 1526 (Fed. Cir. 1994) (en banc).

forms. At one extreme source code may be incomplete and fragmentary and written down on paper. As such it may be primarily communicative—the notes from a brain-storming session. At the other extreme source code may be recorded to disk in a complete and compilable form. Therefore it may be primarily functional and ready to run on a computer. The First Amendment status of each kind of code should be different.

The line between functional and non-functional code is unavoidably somewhat arbitrary. Source code is at least one step away from finished, compiled, fully-functioning object code. But electronic source code that is ready to compile merely needs a few keystrokes to generate object code—the equivalent of flipping an "on" switch.[41] Code used for this purpose can fairly easily be characterized as "essentially functional." The hard cases come when source code is more than one step away from object code, when enablement is not quite as simple. What if two affirmative steps are required to generate object code, or three, or twenty? At what point does source code shift from essentially functional to essentially communicative? At what point does source code shift from being like a constitutionally unprotected machine needing to be flipped on, or perhaps unprotected machine parts ("some assembly required"), to a constitutionally protected recipe or instruction manual? This is a tough question—one which may not admit of a bright line rule. It requires a very fact-based inquiry. But it is the kind of analysis that courts should make.

Wherever the line is drawn, executable code generally can be regulated by the government. This is because such code is usually addressed to *computers*, not people, and therefore does not qualify as speech. The distinction between source code and object code should not necessarily make a difference for this analysis. Consider that programmers write in source code and not object code because it is a happy convenience. High-level languages make it possible to author complex software that otherwise would be far too labor-intensive to create. But such source code is still ultimately addressed to the machine. The only sense in which such code "communicates" is that it enables a general tool (the computer) to do a particular task (say, encrypt data). This is not speech. This is engineering, and it does not merit First Amendment protection.

This argument stands even in the face of the *Bernstein* court's claim that source code is a "language." The *Yniguez* court defined "language" as

41. In this sense, cryptographic source code is like a machine that makes a machine (the object code) that makes locked envelopes.

a "complex system of understood meanings."[42] Of course, the word "understood" implies the existence of someone who understands. But a computer is not something that understands, and so human-to-computer communication is not what the First Amendment is designed to protect.[43] Therefore, at least when addressed to computers, electronic source code is not a language in the sense "German or French" are because it does not involve human-to-human communication.

All of this is not to say that source code is never communicative, and that source code is only addressed to computers. Although one commentator has written that "it is not possible to create communications, thoughts, or ideas in cryptographic source code,"[44] he is wrong. People do communicate to each other in source code, and they publish their ideas about programming in books. It is not seriously disputed that these publications are speech accorded full First Amendment protection.

Nor does the analysis change the result in *Bernstein I.* This comment does not address what happens if source code is thoroughly communicative *and* functional, both language and machine.[45] Even fully functional source code is not always addressed to computers—sometimes people communicate to each other in executable (compilable) source code. Suppose, for example, someone wants to explain to others how a cryptographic computer algorithm should best be written. He might electronically communicate a program, perhaps written in "C," to other people: perhaps his students. This, of course, is exactly what Daniel Bernstein planned to do. The court in *Bernstein I* was therefore correct to analyze Snuffle as speech. This comment is merely a warning not to read the holding too broadly in the future, and does not disagree with the judgment.

IV. CONCLUSION

The court may well have reached the correct result in *Bernstein I* by refusing to enforce the government's licensing schemes regulating the export of encryption software. But in handing down its judgment, the court created a holding too sweeping in scope. Not all source code is speech protected by the First Amendment. Rather, to decide which examples of

42. Yniguez v. Arizonans for Official English, 69 F.3d 920, 935 (9th Cir. 1995) (en banc), *vacated as moot,* __ U.S. __, 117 S. Ct. 1055 (1997).

43. There is no need to delve into the philosophy of mind here—even if one were to argue that a computer can understand, nevertheless it is not a mind guaranteed First Amendment protection. Computer understanding is not something the law cares about.

44. John P. Collins, Note, *Speaking in Code,* 106 YALE L.J. 2691, 2694 (1997).

45. The prior restraint issue, which drove the court's decisions in *Bernstein I & II,* is also not discussed in this comment.

source code deserve First Amendment protection, future courts should engage in a detailed factual analysis of (1) the form in which source code is distributed, and (2) to whom source code is addressed.