RICHARD R. WINTER, ESQ.
SARAH E. PACE, ESQ.
McBride Baker & Coles
500 West Madison Street
Chicago, IL 60661
(312) 715–5778

JAMES WHEATON, ESQ.; SBN 115230
FIRST AMENDMENT PROJECT
1736 Franklin, 8th Floor
Oakland, CA 94612
(510) 208–7744

KARL OLSON, ESQ.; SBN 104760
Levy, Ram, Olson & Rossi
639 Front Street, 4th Floor
San Francisco, CA 94111
(415) 433–4949

ROBERT CORN-REVERE, ESQ.
Hogan & Hartson, L.L.P.
555 Thirteenth Street, NW
Washington, DC 20004
(202) 637–5600

Attorneys for Plaintiff
Daniel J. Bernstein

IN THE UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF CALIFORNIA

DANIEL J. BERNSTEIN, Plaintiff, v. UNITED STATES DEPARTMENT OF COMMERCE, et al., Defendants.	C 95–00582 MHP **DECLARATION OF DANIEL J. BERNSTEIN IN SUPPORT OF PLAINTIFF'S MOTION FOR SUMMARY JUDGMENT** HEARING DATE: August 2, 2002 TIME: JUDGE: Marilyn Hall Patel

I, DANIEL J. BERNSTEIN, hereby declare:

1. I am the Plaintiff in the above-entitled action. I currently reside in Chicago, Illinois. Except as expressly stated below, I have personal knowledge of the facts stated herein. If called upon to testify, I would competently testify to these facts.

2. My statements in this declaration about the effects of the Export Administration Regulations ("EAR") are my descriptions of the literal meaning of EAR. They are not meant as statements about the legal effect of EAR.

Curriculum Vitae

3. From 1991 through 1995, I was a graduate student in the Department of Mathematics at the University of California at Berkeley. I received my Ph.D. in 1995.

4. In summer 1992 and again in summer 1993, I worked with Arjen Lenstra at Bellcore, a research laboratory in New Jersey operated by the major local telephone companies.

5. From 1995 through 1998, I held a temporary faculty position, Research Assistant Professor, in the Department of Mathematics, Statistics, and Computer Science at the University of Illinois at Chicago (“UIC”).

6. For three years beginning in summer 1996, I held the position of Principal Investigator under a research grant from the Division of Mathematical Sciences (“DMS”) at the National Science Foundation. The grant title was DMS–9600083: Algorithmic Problems in Number Theory.

7. Starting in 1998, I held a tenure-track faculty position, Assistant Professor, at UIC. I received tenure in August 2001. I now hold a permanent position, Associate Professor, with the possibility of future promotion to Professor.

8. For three years beginning in summer 1999, I held the position of Principal Investigator under a research grant from DMS. The grant title was DMS–9970409: Algorithmic Problems in Number Theory.

9. From August 2000 through December 2000, I took leave from UIC and held a position of Key Senior Scientist in the Algorithmic Number Theory program at the Mathematical Sciences Research Institute in Berkeley, California.

10. From September 2000 through the present, I have also held the position of Principal Investigator under a research grant from the Division of Computer-Communications Research at the National Science Foundation. The grant title is CCR–9983950: CAREER: Computational Number Theory, Cryptography, and Computer Security.

11. In February 2002, I received a letter from the Alfred P. Sloan Foundation informing me that I have been selected as an Alfred P. Sloan Research Fellow. Exhibit A is a true and correct copy of that letter.

12. On 31 March 2002, I received email from a Program Director in DMS stating her intent to recommend funding for a proposed research grant under which I would be Principal Investigator for three years beginning in summer 2002. The proposal title is DMS–0140542: Algorithmic Problems in Number Theory. Exhibit B is a true and correct printout of that email.

13. I intend to use a combination of research funding and sabbatical leave to replace my regular teaching duties for the next two years with time spent on some urgent research problems. I plan to attend two programs at the Mathematical Sciences Research Institute from August 2002 through March 2003, possibly as a Visiting Scholar at the University of California at Berkeley.

Scientific Conferences

14. A large part of my scientific activity has revolved around formal scientific conferences. I have attended approximately forty formal scientific conferences in the past ten years. (Scientists normally say simply “conference.” I am using the phrase “formal scientific conference” in this declaration to avoid confusion with the broader English meaning of “conference.”)

15. There are two main features of a formal scientific conference. The first is a series of formal lectures. Usually the lecture topic is announced in advance of the conference, but sometimes lectures are prepared at the last moment. A typical lecture lasts 30 or 40 minutes, though lectures can be much shorter or longer. The oral component of a lecture is almost always supplemented by visual displays using projectors or blackboards.

16. Formal lectures are an efficient mechanism to communicate new research results to a large group of people. I have announced many of my own results in lectures at formal scientific conferences, and I have learned a tremendous amount from listening to other lectures.

17. I gave invited lectures at formal scientific conferences at the Oberwolfach and Dagstuhl research centers in Germany in October 1994, May 1995, May 1997, October 1998, July 1999, May 2001, and July 2001; at the Foundations of Computational Mathematics workshop in Hong Kong in October 1999; at the IMACS Conference on Applications of Computer Algebra in Russia in June 2000; at the Durham Symposium on Computational Number Theory in England in July 2000; at various formal scientific conferences in Canada in May 1994, June 1999, June 2000, and October 2001; and at various formal scientific conferences in the United States in December 1995, March 1997, October 1997, September 1998, April 2000, May 2000, August 2000 (three lectures), October 2000, September 2001, and November 2001.

18. The second main feature of a formal scientific conference is a series of opportunities for a scientist to engage in highly interactive discussions with his colleagues. These discussions account for a large part of my time at formal scientific conferences, and they are one of the main reasons I attend.

19. I expect, as a technological matter, that a similar level of interactivity will eventually be achieved for online communication among groups of users, with each user having an extremely fast Internet connection, appropriate software, a microphone, a video camera, and a gigantic computer display that can simultaneously show the gestures and facial expressions of other users and several pages of shared information. However, as far as I know, none of my colleagues have these tools, while a very large number of them show up in person at formal scientific conferences.

20. Scientists can also visit each other outside the context of a formal scientific conference. For example, I have visited my colleague Jonathan Sorenson to work on a joint research project; our collaboration involved much faster interaction than would have been physically possible through email.

21. Formal scientific conferences have several collaborative advantages over visits: they bring many people together simultaneously; they allow unexpected interactions to

develop on the fly; and they allow short interactions that alone would not have justified the effort of travel.

22. Whenever a conference approaches, I start “ramping up” for it: thinking about who I will see, considering questions I might ask, considering questions I might be asked, reviewing potentially relevant material, completing relevant experiments, etc. I continue these preparations on the way to the conference, aiming to make the conference time as productive as possible.

23. I have been invited, and plan, to give a lecture at the Symposium on Cryptography in another formal scientific conference, the Summer Meeting of the Canadian Mathematical Society at the University of Laval, Quebec, in June 2002. I also plan to attend several formal scientific conferences this year without giving formal lectures: the Illinois Number Theory Conference in Urbana in May 2002, the USENIX Security Symposium in San Francisco from 5 August 2002 through 9 August 2002, the CRYPTO conference in Santa Barbara from 18 August 2002 through 22 August 2002, and several subsequent conferences in Berkeley.

Publications

24. I now use Internet web pages as the preferred, though not the only, medium for publishing my work. I have written, and published through my web server at http://cr.yp.to, more than two hundred thousand lines of text in various forms.

25. I wrote a program in December 2000 to keep track of changes to my web pages. Between 22 December 2000 and 4 April 2002, there were more than three thousand occasions on which I either added a file to my web pages or modified an existing file on my web pages.

26. I sometimes publish, through my web server, copies of documents that I have found elsewhere on the web. For example, I frequently publish copies of documents relevant to my courses, so that my students can retrieve the documents even when UIC’s Internet connection is overloaded.

27. The practice of making a document available from several computers around the Internet is called “mirroring.” The extra copies of the document are called “mirrors.”

Changes to the original document are not always immediately reflected in the mirrors, despite the terminology, but they are usually reflected within a short time.

28. Mirroring is, as a technological matter, essential for extremely popular documents, because there are limits to the number of users who can simultaneously download a document from a single computer. Ten mirrors can handle approximately ten times as many users.

29. Even when it is not essential, mirroring provides many benefits to Internet users. For example, users can download a document much more quickly and reliably if there is a nearby mirror, as illustrated by the example of my students at UIC.

Research Motivation, Part One: Sniffers

30. Most of my research is motivated, directly or indirectly, by the critical problem of protecting the Internet against attack. I worry about attackers ranging from petty thieves stealing credit-card numbers to cyber-terrorists disrupting the entire World Wide Web.

31. One way that Internet attackers gain unauthorized access to computers is by monitoring and imitating the Internet activities of authorized users. For example, if a legitimate user's password is sent through an Internet connection monitored by an attacker, the attacker can then use that password.

32. Cryptography provides a framework for preventing this type of attack. Legitimate users have "secret keys" that they use to scramble their messages. The scrambling method is called a "cryptographic system." One hopes that an attacker who eavesdrops upon the scrambled messages cannot "break" the system: figure out the secret keys, for example, or figure out the original messages, or forge a message without detection.

33. Some cryptographic systems are guaranteed to be unbreakable. For example, a 40-digit Gilbert-MacWilliams-Sloane system guarantees that the receiver has a 99.99999999999999999999999999999999999999% chance of detecting each forgery, no matter what forging method the attacker uses. Certain special features of the Gilbert-MacWilliams-Sloane system allow mathematical analysis of all possible forging methods.

34. Unfortunately, protecting all Internet communications with the Gilbert-MacWilliams-Sloane system, or with any other guaranteed system known to me, would be extremely inconvenient. In these systems, users who want to communicate have to meet privately to create a secret key. Furthermore, the secret key can be used for only a limited number of messages, after which the users have to meet again.

35. In contrast, in “public-key” cryptographic systems such as the Diffie-Hellman system, a user can simply create his own secret key, not shared with anyone else. He can then communicate with any number of users. There are no limits on the number of messages.

36. Unfortunately, there are no guarantees of unbreakability for public-key systems. For example, a 5000-digit Diffie-Hellman system is unbreakable by every method I know, but it might be broken by a method published tomorrow. The cryptographic literature has many examples of methods that remained unbroken for years and were then suddenly broken.

37. Furthermore, most public-key systems are not fast enough to be used for all Internet communications. If they were deployed on a busy Internet server, the server would be overwhelmed by cryptographic work.

38. One way to save cryptographic time is to choose a “risky” system that is close to, but not quite, breakable by current attacks. These systems are typically much faster than “conservative” systems. However, this strategy relies on having an extremely precise idea of how far the attacks can be pushed.

39. One of my research projects is to determine the exact costs of breaking various systems. The goal is not to help attackers actually break the systems; the goal is to understand which systems can be broken, so that users can switch to safe systems. It is essential for public knowledge to stay ahead of what the criminals have figured out.

40. Another of my research projects is to speed up cryptographic computations without losing security: in part by finding faster methods of performing the computations in existing systems, and in part by finding better systems.

Research Motivation, Part Two: Bugs

41. Another way that Internet attackers gain unauthorized access to computers is by locating and exploiting mistakes in Internet software.

42. For example, a programming mistake in versions 8.8.0 and 8.8.1 of a program called “Sendmail” allowed any attacker on the Internet who knew about the mistake to seize control of any computer using those versions of Sendmail to handle Internet mail. I know this from reading the relevant portions of the program.

43. The programming mistake in Sendmail versions 8.8.0 and 8.8.1 is an extremely common type of mistake called “unchecked array access.” The attack is called a “buffer overflow.” I have seen hundreds of public reports of unchecked array accesses allowing attackers to seize control of various programs through buffer overflows. I have personally verified enough of the reports to be confident that most of them are accurate.

44. There are many other common programming mistakes allowing Internet attackers to gain unauthorized access to various programs. On the basis of my review of many public reports, I estimate that, at every moment during the past three years, millions of computers around the Internet were using software with serious security problems.

45. An attacker who seizes control of a computer can inspect, modify, and destroy information on that computer. He can, for example, steal copies of any secret keys stored on that computer, nullifying the cryptographic protection provided by those keys.

46. Many exploitable mistakes in specific programs have been found and fixed by the security community. However, I estimate that finding all the mistakes in current software will take decades. Even worse, exploitable mistakes continue to appear in new software and in modifications to old software.

47. One of my research projects is to change widespread software-writing practice so that common programming mistakes do not lead to security problems. It is not enough to design “fail safe” programming techniques for my own use; the techniques need to be so convenient that novice programmers will happily use them.

48. Another of my research projects is to identify and fix all of the current problems. Software plagued with mistakes also tends to be hard to read and modify, so writing an entirely new replacement is often easier than fixing the mistakes. As in cryptography, the goal of identifying problems is not to help attackers exploit the problems, but to help users select safe systems.

Research Motivation, Part Three: Floods

49. Internet attackers sometimes carry out a third type of attack, flooding a server with more work than the server can handle, to deny service to legitimate users. Flooding the Internet ".com servers," for example, would effectively disable the World Wide Web.

50. Flooding is particularly easy for an attacker who has gained unauthorized access to tens of thousands of computers, for example through the aforementioned "buffer overflows." All those computers can join forces in flooding the server.

51. One of my research projects is to redesign the Internet's public-communication protocols so that flooding a central site does not destroy service for the entire Internet. Another of my research projects is to redesign the Internet's private-communication protocols so that parties who know each other can continue communicating even during a high-volume flood. High-speed cryptography is crucial: the attacker's forgeries must be distinguished from trusted messages and discarded as quickly as possible.

Research Motivation, Part Four: Other Applications

52. My research has additional motivations beyond Internet security. Fifteen of my research projects have set public world speed records for the computation of various mathematical functions; all of these functions have applications outside Internet security.

53. One extreme example is the "discrete Fourier transform," a fundamental mathematical function relevant to cryptography but also useful for image processing, seismography, and many other applications. In 1997, I set a new world speed record, by a factor of 2, for discrete Fourier transforms on the Pentium computer architecture.

54. At another extreme is "integer factorization." This mathematical function is described in many books only as a way for attackers to break various public-key cryptographic

systems. Within mathematics, however, this function has many more applications. For example, mathematicians often use integer factorization to help compute “integral closures,” which are important objects of study in pure number theory.

55. Several of my speed records are for various functions arising inside integer factorization. I would have done this work even if it had no applications aside from helping users select safe cryptographic systems; however, I also value the other applications.

56. A few of my projects are motivated entirely by applications outside Internet security. For example, I wrote software for computing solutions to “diagonal equations” when one of my colleagues, Yuri Tschinkel, asked me for numerical examples related to certain mathematical objects called “diagonal cubic surfaces.”

Software Example: qmail

57. Programs called “message transfer agents” (“MTAs”) play the role of post offices in the Internet mail system. They accept mail from local users and send the mail to other MTAs. They also receive mail from other MTAs and deliver the mail to local users.

58. On several occasions, first in November 1996 and most recently in October 2001, I have surveyed the MTAs used to receive Internet mail. In my surveys, I select many Internet addresses at random, attempt to connect to MTAs at those addresses, and ask the MTAs to identify themselves.

59. In my November 1996 survey, 80% of the MTAs identified themselves as various versions of the aforementioned Sendmail program. I had similar impressions of Sendmail’s popularity long before carrying out the survey: my own mailboxes at various universities were handled by Sendmail, and Sendmail was very widely discussed in public.

60. In 1995, after hearing some new reports of exploitable mistakes in Sendmail and reviewing portions of the Sendmail program, I concluded that fixing Sendmail would be practically impossible. I investigated several other MTAs and eventually decided to write my own, qmail, with security as the top priority.

61. I published the first usable draft of qmail early in 1996, along with a prediction that there were many more exploitable mistakes waiting to be discovered in Sendmail. I saw

fourteen public reports of Sendmail security problems in 1996 and 1997; I stopped counting at that point.

62. I have continued working on qmail and several related programs. In March 1997, following suggestions from several people, I took the unprecedented step of offering a $500 security reward to the first person to publish a verifiable security hole in qmail. Nobody has claimed the reward.

63. In my October 2001 survey, 17% of the MTAs identified themselves as qmail, while Sendmail had dropped to 42%. I am comforted by the thought that nearly a million computers are using qmail and will be immune to the next Sendmail security problem.

64. Like any other program, qmail is a sequence of instructions. I have seen messages from many people providing modifications (“patches”) to, and other comments about the details of, these instructions. These messages have led to many improvements in qmail. For example, Exhibit C is a true and correct printout of a message sent to my public qmail mailing list on April 1, 2002. It displays a modification to “rblsmtpd,” a program that I published for use along with qmail. I have decided to make a similar modification when I prepare the next draft of rblsmtpd.

Software Example: tinydns

65. Internet computers have “IP addresses.” IP addresses are analogous to phone numbers. If your computer wants to reach www.citysearch.com, for example, it needs the IP address for www.citysearch.com, currently 209.104.39.15.

66. IP addresses are published by programs called “domain-name servers.” To find the IP address for www.citysearch.com, your computer first contacts one of the aforementioned “.com servers.” (There are only about a dozen .com servers on the entire Internet, and your computer already has their IP addresses.) The .com servers tell your computer the IP addresses of the “.citysearch.com servers.” Your computer then contacts one of the latter servers to learn the address of www.citysearch.com.

67. A particular software package called “BIND” consists of a domain-name server and several related programs. I am under the impression that BIND was used for practically

all the domain-name servers on the Internet for many years: I saw many computers using it, and none using anything else.

68. In July 1999, in the “Secure replacements for architecturally insecure programs” section of my CCR–9983950 proposal to the National Science Foundation, I discussed the Sendmail/qmail history and then said that my next project was to replace BIND. BIND’s security record was not as bad as Sendmail’s, but it was clear to me from reading the BIND software that BIND was fundamentally flawed.

69. In November 1999, I saw a public report of an exploitable mistake in BIND. Around Christmas 1999, I published the first usable draft of a BIND replacement, including a new domain-name server, “tinydns.” I have continued working on this software since then. In January 2001, I saw a public report of another exploitable mistake in BIND.

70. I estimate that more than four million .com domains, including citysearch.com, are now published through tinydns. This estimate is based on surveys with more room for error than the aforementioned MTA surveys; I would not be surprised if the correct number is anywhere between two million and eight million.

71. As with qmail, I have seen and benefited from messages from many people providing patches to, and other comments about the details of, this software.

Software Example: hash127

72. In 1999, I published software called “hash127” showing how a particular mathematical function could be computed several times more quickly than was previously believed.

73. The primary application of hash127 is protecting messages against forgery. There are many other mathematical functions used for that purpose, but hash127 was so fast that it set new speed records for message authentication at its security level. The improvement varied with message length and with computer architecture but was often above 50%.

74. Speedups in message authentication are directly relevant to Internet security. Every 10% speedup means, for example, another 10% increase in the volume of a flood that a computer can handle while continuing to communicate with trusted parties.

75. The computational technique I used in hash127 has several variants: a “64-bit method” and a “53-bit method” and a “10-digit method” and so on. These variants refer to the precision of the numbers used throughout the calculation.

76. For example, if someone wants to compute this mathematical function by hand with the help of a calculator with a 10-digit display, he will find it most efficient to use the 10-digit method. He will find it inconvenient to use the 11-digit method. He will find it convenient to use the 9-digit or 8-digit method, but the 10-digit method is faster.

77. Modern general-purpose computers contain a fast 53-bit calculator. Intel-compatible computers such as the Pentium and Athlon also contain a fast 64-bit calculator. When I was writing hash127, I started with the 64-bit method, for top speed, and then wrote the 53-bit method, to cover non-Intel-compatible computers.

Software Example: nistp224

78. At some point I started writing software to compute another mathematical function, the “NIST P-224 elliptic-curve Diffie-Hellman key-exchange function” (“P-224 function”), using the techniques I had introduced in hash127.

79. The P-224 function, like the Diffie-Hellman key-exchange system generally, has two primary applications: protecting messages against eavesdropping, and protecting messages against forgery. There is no difference between computing the function for one application and computing it for the other. The result of computing the function is a number useful for both applications.

80. I finished writing software for the P-224 function using the 64-bit method at the end of August 2001. I did not publish it at that point; it was not complete without the 53-bit method.

81. I finished writing the complete software, “nistp224,” on 26 September 2001. The result was a set of new public world speed records for the entire Diffie-Hellman key-exchange system at that security level: for example, 3 times faster on one common type of computer.

82. On 29 September 2001, after consulting with my attorneys, I made the nistp224 software publicly available from my web pages. To protect myself from prosecution, I sent an

email message to crypt@bxa.doc.gov ten seconds before this publication. Exhibit D is a true and correct printout of the email message.

Software Example: SPRAY

83. On various occasions I have experimented with certain mathematical functions called “pseudorandom number generators” (“PRNGs”). PRNGs are widely used to protect messages against forgery; they can, for example, be combined with the aforementioned P-224 and hash127 functions for this purpose. PRNGs are also widely used to protect messages against eavesdropping. PRNGs are also widely used for many applications having nothing to do with information security.

84. A PRNG designed to protect messages against forgery can also be used for other applications, and in particular to protect messages against eavesdropping. Using it in this way is just as easy as using a PRNG that was designed to protect messages against eavesdropping. Another type of mathematical function used in cryptography, called a “cryptographic hash function,” can also be used as a PRNG suitable for all of these applications, with only slightly more effort.

85. However, a PRNG designed to protect messages against eavesdropping is typically several times faster. As noted above, high-speed cryptography is crucial for protecting the Internet.

86. Exhibit E is a true and correct printout of “SPRAY,” some fast pseudorandom number generation software that I wrote a few years ago, specifically for protecting messages against eavesdropping. This document is dated 2 July 1998 inside my computer.

87. I wrote SPRAY in a language called “x86 assembly language.” Assembly languages are particularly well suited for high-speed computation. Writing a program in assembly language, rather than in another language, is analogous to providing an instruction video, rather than an instruction sheet, for putting together a piece of furniture: the video takes more space, but it shows you exactly how you should move your hands at every moment, and can also show you techniques that are difficult to express on paper.

Software Example: Snuffle

88. About twelve years ago, as an undergraduate at New York University, I heard that the government regulated exports of anti-eavesdropping software, but that it permitted exports of cryptographic hash software, such as Ralph Merkle's Snefru.

89. This struck me as silly. I did not know nearly as much about cryptography then as I know now, but I did know that typical cryptographic hash methods relied on the same techniques as typical methods for protecting messages against eavesdropping.

90. To prove the point, I wrote a short program, "snuffle.c" (AER 4). This program consists of a few instructions in the C language that, when used together with Snefru, scramble messages to protect them against eavesdropping. I wrote a companion program, "unsnuffle.c" (AER 5), to undo the scrambling.

91. I also wrote a short paper, "The Snuffle Encryption System" (AER 3), with various information in a combination of English and mathematical notation.

92. In a letter dated 30 June 1992 (AER 18), I asked the government for permission to publish the programs along with the paper. I commented that I wanted to publish these items for discussion by the worldwide academic community. I also commented that I foresaw Snuffle being used in practice.

93. I no longer foresee Snuffle being used in practice. Modern methods such as SPRAY are substantially faster.

94. In a letter dated Aug 20 1992 (AER 20), the government denied permission: "REQUEST FOR COMMODITY JURISDICTION DETERMINATION FOR: Snuffle 5.0 Software . . . the referenced commodity is subject to the licensing jurisdiction of the Department of State . . . This commodity . . . is designated as a defense article."

95. At some point it occurred to me that the government might be trying to draw another line that I considered silly: namely, a line between the programs and the paper.

96. I prepared a set of five documents: DJBCJF-2, DJBCJF-3, DJBCJF-4, DJBCJF-5, and DJBCJF-6. DJBCJF-2 was the paper. DJBCJF-3 was snuffle.c. DJBCJF-4 was unsnuffle.c. DJBCJF-5 (AER 6) was a sequence of English instructions that, when

followed, would scramble messages the same way as snuffle.c. DJBCJF-6 (AER 7) was a sequence of English instructions that, when followed by a programmer, would produce a C program that had the same effect as snuffle.c.

97. In a letter dated 15 July 1993 (AER 38), I asked the government for permission to publish DJBCJF-2. I sent four additional letters the same day making identical requests for DJBCJF-3 (AER 37), DJBCJF-4 (AER 36), DJBCJF-5 (AER 35), and DJBCJF-6 (AER 34).

98. My primary objective in submitting these requests was to see whether the government would assert control over the paper, DJBCJF-2. Otherwise they would be forced to draw a line on the DJBCJF-2, DJBCJF-5, DJBCJF-6, DJBCJF-3 spectrum; I planned, in that event, to file further requests to continue narrowing the gap down to something that would strike even the most casual observer as being absurd.

99. In a letter dated Oct 5 1993 (AER 40), the government refused permission, designating all the documents as defense articles: "REQUEST FOR COMMODITY JURISDICTION FOR: DJBCJF-2, DJBCJF-3, DJBCJF-4, DJBCJF-5, DJBCJF-6 . . . the referenced items are subject to the licensing jurisdiction of the Department of State . . . The referenced items . . . are designated as defense articles."

100. I filed this lawsuit in February 1995. In a response dated Jun 29 1995 (AER 42), the government suddenly started drawing lines. It asserted that DJBCJF-5 and DJBCJF-6 were "technical data" and that my paper did not appear to be controlled at all.

Software Example: dh224

101. Several years ago, I combined a few cryptographic techniques into a software package, "dh227," that allowed other programs to easily protect messages against eavesdropping and forgery.

102. I was not satisfied with dh227. It solved one of the basic speed problems of public-key cryptography, but it was still not nearly fast enough for high-volume communication. Furthermore, it did nothing to prevent floods and other denial-of-service attacks.

103. Exhibit F is a true and correct printout of my current draft of a newer and faster software package, “dh224,” designed to allow other programs to easily protect messages against forgery, with high resistance to floods.

104. I am extremely uncomfortable to be revealing such a raw draft. The software needs to cleaned up, made much easier to read, analyzed for mistakes, and analyzed for cryptographic strength. It would be scientifically irresponsible for me to put dh224 on my web pages at this point; there could be a major error making forgeries easy.

105. However, this draft does let other programs communicate in the desired fashion. I expect to have the remaining work done, and to have dh224 ready for formal publication on my web pages, before my lecture at the aforementioned Symposium on Cryptography in Canada in June 2002.

106. There are several further modifications that I plan to make to dh224. For example, I plan to integrate a variant of hash127 into dh224, along with a good PRNG, to speed up message authentication. I am considering SPRAY as the PRNG; even though I designed SPRAY for another application, it is faster for this application than anything else I have seen.

107. By adding a few additional lines to dh224, I could turn the protection against forgery into protection against both forgery and eavesdropping. As was previously illustrated by Snuffle, the differences are, from a cryptographer’s perspective, superficial.

Impact of the Current Regulations

108. EAR, despite the January 2000 improvements, still prohibits my desired activities in seven situations. I am refraining from those activities, and I am frequently refraining from getting myself into those situations in the first place.

109. For nearly two years, from January 2000 through November 2001, I put extensive effort into expressing my remaining concerns to the defendants, trying to understand their goals, trying to see if we could settle the case, and trying to have the regulations fixed. The parties exchanged many detailed communications, mostly in the context of formal settlement discussions.

110. In recent case-management discussions with this Court, the defendants repeatedly pointed to the gap between a March 2001 deadline set by the Court and a July 2001 filing from my attorneys. In fact, we were embroiled in settlement discussions throughout that gap. My records indicate that there were written communications between my attorneys and the defendants' attorneys on each of the following dates during that period: 5 March 2001, 9 March 2001, 14 March 2001, 28 April 2001, 6 May 2001, 9 May 2001, 8 June 2001, 19 June 2001, 2 July 2001, and 10 July 2001.

111. My efforts in 2000 led to one success. In February 2000, the defendants sent me a letter in response to some questions that I had posed. Exhibit G is a true and correct copy of that letter. The statement "[b]inary code which is compiled from TSU source code and which is itself publicly available and not subject to licensing or royalty fee can also be exported under the provisions of license exception TSU" in that letter contradicted the plain meaning of the regulations at that time; but, in October 2000, the defendants modified EAR, adding EAR §740.13(e)(2), making the regulations consistent with that statement.

112. However, as far as I know, there have been no other helpful changes in EAR. I remain fearful of criminal prosecution and civil penalties under EAR.

Problem One: Collaboration at Scientific Conferences

113. EAR prohibits me from working collaboratively with my foreign colleagues on various types of software at a formal scientific conference.

114. The problem here is the clear requirement in EAR §740.13(e)(1) that notification to the government take place "by the time of export." There are three levels of difficulty in trying to comply with this requirement when I am working with someone in person.

115. First, two people working together on software are continually creating, and disclosing to each other, new versions of the software. Even if we are working on a computer with Internet access, it would dramatically slow down the collaboration to have me stop before every change to hide the computer screen from the other person, make the change, send a copy to the government, and then reveal the screen again.

116. Second, the formal scientific conferences that I attend almost never have Internet access in the conference rooms. Furthermore, at many conferences, there are only a few Internet connections available for all the conference participants.

117. Third, at some conferences, Internet access is practically impossible. On several occasions I have spent entire week-long trips unable to check my email.

118. I see no reasonable way to take advantage of EAR §740.13(e)(1) at even the first level of difficulty, let alone the second or the third.

119. Consequently, at formal scientific conferences in the United States, to obey EAR §764.2(e), I am refraining from disclosing any new “EI” software to my foreign colleagues. I am aware, far beyond the level of “knowledge” defined in EAR §772, that many of my foreign colleagues ignore the regulations.

120. At formal scientific conferences in Canada, to obey EAR §736.2(b)(1) and EAR §736.2(b)(2), I am refraining from showing my foreign colleagues any “EI” software that I wrote on the way to the conference, and anything based on that software.

121. Furthermore, at formal scientific conferences anywhere outside the United States, to obey EAR §744.9(a), I am refraining from working on “EI” software with my foreign colleagues.

122. Here is an example. On 22 January 2001, I was invited to give a lecture at the Elliptic-Curve Cryptography (“ECC 2001”) conference at the University of Waterloo in Canada. The conference took place at the end of October 2001.

123. On 28 October 2001, I took the train to Canada for ECC 2001. I used the travel time to work on my laptop computer, “ramping up” for the conference as usual. One of the documents I worked on during the trip was ProSPRAY, an experimental offshoot of SPRAY.

124. I was unable to connect my laptop computer to the Internet on the way to ECC 2001. I was also unable to connect my laptop computer to the Internet at ECC 2001.

125. More than 100 people attended ECC 2001. I believe that many of the participants were from outside the United States. I also believe that at least two of the participants, Brian Snow and Jerome Solinas, were National Security Agency employees.

126. I had various discussions at ECC 2001 with scientists whom I believe to be from outside the United States. For example, I had various discussions with Robert Harley, who I believe is an Irish citizen working in France.

127. At one point, Mr. Harley and I were comparing various features of the Compaq Alpha and Pentium Pro computer architectures. It occurred to me that Mr. Harley would probably have useful comments on the suitability of ProSPRAY for the Alpha, and that we would probably be able to collaboratively put together an improved version suitable for both architectures. However, because of EAR, I refrained from proceeding down this path, and from engaging in various other collaborative activities at ECC 2001.

128. I expect to be in similar situations at the aforementioned meetings in Urbana in May 2002, in Quebec in June 2002, in San Francisco in August 2002, etc. In the absence of court protection or a change in the regulations, I will be forced to continue sacrificing opportunities for collaboration with foreigners at conferences.

Problem Two: Private Email

129. EAR demands disclosure of my email when that email contains certain types of information and is sent to a foreign colleague. I do not like this, because it violates my privacy.

130. There are two types of disclosure at issue here. The first is disclosure to the government; this is required whenever EAR demands notification or licensing. The second is disclosure to the public; this is required whenever EAR demands “public availability,” if the information is neither “fundamental research” nor “educational.”

131. To avoid these situations, I have generally refrained from private discussions related to “EI” information, except when I know that the recipient is a United States citizen.

132. There are three reasons that I value the privacy of my email. First, disclosure can lead to misuse. If, for example, I am discussing an exciting new security system with my colleagues, an innocent user sees it and starts using it, and our analysis subsequently reveals that it has a horrible flaw, then the user’s computer may already have been penetrated. Part of a scientist’s responsibility is to keep overly raw ideas out of the public eye until their merit has been assessed.

133. Second, even when the information is not misused, its disclosure can be embarrassing. I feel much more free to experiment with wild ideas in private discussions with my trusted colleagues than in public; those discussions should not be revealed to anyone else.

134. Third, even when the information is not misused and not embarrassing, I do not want to be forced to disclose it. The forced disclosure is emotionally distressing.

Problem Three: Web Publication

135. EAR demands notification when I place certain types of information on my web pages. The problem here is the amount of time spent figuring out exactly which of my documents the government is demanding to see.

136. My web pages include many documents that focus on something other than cryptography but that really should include cryptographic pieces. For example, I would like to include cryptographic components in qmail, tinydns, and many other programs. I have refrained from doing so, because I do not want to incur the cost of trying to review each of these changes from the government's point of view.

137. There are several reasons that I find it difficult to figure out which of my documents require notifications under EAR. First, EAR does not specify the circumstances under which a function useful inside an "encryption function" is itself an "encryption function." It is, for example, unclear whether the P-224 function, whose applications are balanced between anti-forgery and anti-eavesdropping, is an "encryption function." It is also unclear to what extent a good PRNG is necessarily an "encryption function." It is also unclear whether Snefru and Snuffle, which can be combined to scramble data, are "encryption functions." It is also unclear whether a discrete Fourier transform is an "encryption function": Fourier transforms are useful for integer multiplication, which in turn is useful for "modular exponentiation," which in turn is useful for several cryptographic systems.

138. Second, EAR does not specify the circumstances under which software that includes "EI" software is itself "EI" software. For example, dh224 protects messages against forgery but not eavesdropping; however, it uses nistp224, and a subsequent version could use

SPRAY. It is unclear whether including nistp224 or SPRAY inside dh224 would make dh224 "EI."

139. Third, it is not clear what is covered under EAR's notion of being "[d]esigned or modified" for non-authentication functions. I believe, on the basis of reports such as the declaration of John Liebman (AER 500), that the government asserts control over all items that *could* be modified for such functions. It is unclear whether dh224 and other anti-forgery programs are therefore "EI."

140. Fourth, EAR defines "program" quite broadly as any sequence of instructions "in, or convertible into, a form executable by an electronic computer." A programmer can easily convert typical English instructions and mathematical notations into C, so those are "programs" too, and they need review. It is not clear, for example, which of my Snuffle-related documents constitute "programs."

141. An additional problem arises in mirroring. I am much less familiar with the documents I mirror than with the documents I wrote myself. Determining their status under EAR would take considerable extra time. I have drastically limited my mirroring to avoid this situation.

142. I have already spent more than 500 hours trying to figure out what the government wants to see. I anticipate a continuing cost of more than fifty hours per year to review my web pages for EAR notifications, if I start making all the changes that I want to make.

143. I could have my computer send the government a notice of *every* change to my web pages. However, if everyone did this, the government would be flooded with email. I am fearful that the government would attempt to prosecute me under EAR §764.2(h).

Problem Four: Assembly Language

144. EAR requires a license for publication of "EI" software written in any assembly language, such as x86 assembly language.

145. Assembly-language programs are becoming an increasingly large part of my work. As noted above, assembly languages are particularly well suited for high-speed

computation, which in turn is vital for cryptographic protection of the Internet. I would like to rewrite hash127, nistp224, et al. in assembly language.

146. The problem here is that the definitions of “encryption source code” and “encryption object code” in EAR are, together, somewhat less broad than the definition of “encryption software.” In particular, they do not include assembly-language programs, because assembly-language programs are not “compiled.”

147. Consequently I have refrained from publishing assembly-language “EI” software, such as SPRAY.

148. I could artificially build a compiled language with the same expressiveness as assembly language, and translate SPRAY into that language. However, the same technique would allow any program to be treated as “source code” under EAR, escaping the government’s licensing system for “object code.” I am again fearful that the government would attempt to prosecute me under EAR §764.2(h).

Problem Five: Answering Questions

149. EAR requires a license for publicly helping people write “EI” software outside the United States, or privately helping people write “EI” software in any country.

150. Here is an example. There have been approximately two hundred thousand articles published in the “sci.crypt” Internet newsgroup, including a huge number of questions, many of which appear to be from people around the world writing various types of encryption software.

151. In 1997, I wrote an Introduction to Cryptography set of web pages for the students in my cryptography course at UIC. Pursuant to a stipulation with the Secretary of Commerce, I was able to make the first version of my Introduction to Cryptography available to the students in that course, although not to the public.

152. I then expanded the set of web pages to answer many questions that I had seen on sci.crypt. I have continued working on my Introduction to Cryptography since 1997, improving the exposition and adding material to answer more questions. I would put more

time into it if I thought that it could be lawfully published. Exhibit H is a true and correct printout of the current version.

153. Since 1997, I have wanted to publish my Introduction to Cryptography, for the benefit of everyone interested in cryptography, and in particular for the benefit of all people asking these questions on sci.crypt.

154. To obey EAR §744.9(a), I have refrained from publishing my Introduction to Cryptography. I have also refrained from posting immediate answers to such questions unless the questioner was clearly in the United States.

155. Furthermore, to obey the regulations on “export” of “technology,” I have refrained from sending private email answering such questions, whether or not the questioner was in the United States.

Problem Six: sci.crypt and Iran

156. EAR requires a license for me to post “EI” software to sci.crypt.

157. I am aware, far beyond the level of “knowledge” defined in EAR §772, that articles posted to sci.crypt are automatically sent to a huge number of computers, including computers at several universities in Iran.

158. My knowledge was established by a brief web search that I performed early in January 2000, after I heard about the Iran prohibition in the upcoming revision of EAR. In the web search, I located the web page of a system administrator in Iran who obviously read sci.crypt. I also found the names of some university computers in Iran attached to USENET, the portion of the Internet that distributes newsgroups.

159. Consequently, to obey EAR §740.13(e)(3), I have refrained from posting “EI” software to sci.crypt.

Problem Seven: Journals on the Internet

160. EAR generally requires a license for me to publish “EI” software or “technology” through a typical scientific journal or anything else published by a publishing house. There is an exception for printed materials, but most of the journals that I use are also

published through the Internet. There is an exception for “fundamental research,” but I also publish non-research work, such as introductory surveys aimed at graduate students.

161. The problem here is EAR’s extremely narrow definition of “published” in EAR §734.7 and EAR §734 Supplement 1. Most of the journals that I use are priced above the limit set in EAR’s definition of “published.”

162. Consequently I am refraining from preparing non-research “EI” work for journal publication.

I declare under penalty of perjury under the laws of the United States that the foregoing is true and correct and that this declaration was executed on this 26th day of April, 2002.

____
DANIEL J. BERNSTEIN