Grayson Barber (GB 0034)
Grayson Barber, L.L.C.
68 Locust Lane
Princeton, New Jersey 08540
(609) 921-0391

Frank L. Corrado (FLC 9895)
Rossi, Barry, Corrado & Grassi, P.C.
2700 Pacific Avenue
Wildwood, NJ 08260
(609) 729-1333

Attorneys for Plaintiffs

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF NEW JERSEY

EDWARD W. FELTEN; BEDE LIU; SCOTT A. CRAVER; MIN WU; DAN S. WALLACH; BEN SWARTZLANDER; ADAM STUBBLEFIELD; RICHARD DREWS DEAN; and USENIX ASSOCIATION, a Delaware non-profit non-stock corporation, Plaintiffs, vs. RECORDING INDUSTRY ASSOCIATION OF AMERICA, INC.; SECURE DIGITAL MUSIC INITIATIVE FOUNDATION; VERANCE CORPORATION; JOHN ASHCROFT, in his official capacity as ATTORNEY GENERAL OF THE UNITED STATES; DOES 1 through 4, inclusive, Defendants.	Hon. Garrett E. Brown Case No. CV-01-2669 (GEB) Civil Action **DECLARATION OF ANDREW W. APPEL**

I, ANDREW W. APPEL, being of full age, hereby declare and state:

1. I am not a party to this action. Unless otherwise stated, I have personal knowledge of the facts set forth in this Declaration. ~~I make this Declaration in support of Plaintiffs' motion for a preliminary injunction in this matter.~~

2. I am currently Professor of Computer Science at Princeton University. I earned my PhD in Computer Science at Carnegie Mellon University. I have published two books and more than 50 journal and conference papers on topics in programming languages, compilers, and computer security, and I have taught undergraduate and graduate courses in these and other topics for more than 15 years. My full vita is attached as Exhibit A.

3. **Analysis and synthesis:** Good engineering research is a combination of analysis and synthesis. Analysis: a researcher examines an existing system, designs an experiment to measure the system, gets quantitative results, analyzes the results to figure out what they can tell us about its performance, and publishes the results. Synthesis: the researcher designs a new system with better performance, and publishes an explanation of the design. In each case, publication is essential to scientific progress. A researcher who does brilliant science but doesn't explain the results might as well spend his time watching soap operas, for all the good he does society.

4. For example, the 1980's saw a revolution in the design of computers. Professors John Hennessy of Stanford University and David Patterson of Berkeley invented a new analysis of computer architectures. Instead of asking, “how beautiful and symmetric are the instructions” they measured quantitatively “ which instructions do computer programs execute most frequently?” The results of this empirical measurement led them to design “Reduced Instruction Set Computers,” sparking industry-wide improvements in computer performance that continue to this

day. The papers they published -- both the analysis papers and the synthesis papers -- are now classics of computer science.

5. Synthesis without analysis is not great science. Much of the work in computer architecture just before Hennessy and Patterson suffered from this problem: computer engineers did not understand how to best analyze the consequences of their design decisions.

6. Good research in computer security also requires analysis and synthesis. Analysis: "How easy is it to break into this system?" Synthesis: "Let's design a more secure system." Designing computer-security systems without an understanding of how to measure their effectiveness will inevitably lead to weak designs. And a researcher who does brilliant analyses must publish the results, otherwise the work cannot be useful to other scientists.

7. **Modes of publication:** Academic computer scientists normally publish their work so as to reach as wide an audience as possible. For example, like most computer scientists I typically start by writing a "technical report" and putting it on my web page. Then I condense this down to a 10-12 page paper and send it to a conference. Computer science is unlike most disciplines in that conference publication is the most important venue for publication; a good conference will receive 200 submissions and accept fewer than 30 for presentation and for publication in the printed proceedings. Because of the strict length limits on conference papers, the proceedings version often cites the longer technical report (available from my web page) where the interested reader may find more details. Finally, after the conference, I prepare a longer, revised article for publication in a scientific journal.

8. Often, the research leading to a scientific result requires writing a computer program. In order to make my results most useful to their intended audience, I make the programs available

along with the formal scientific paper. In rare cases the program is short enough to be included in the paper itself:

?? "Iterated Register Coalescing" by Lal George and Andrew Appel (ACM Transactions on Programming Languages and Systems, May 1996), is a 20-page paper containing five pages of pseudo-code giving the exact details of the algorithm.

?? "Intensional Equality ;=) for Continuations" by Andrew W. Appel (ACM SIGPLAN Notices, February 1996), a 3-page paper containing a complete 42-line C program explaining how to cheat on benchmark measurements. Although I wrote this paper mostly for my own amusement, I have been told that professors are assigning it as reading in undergraduate classes because it concisely explains the scientific concept.

?? "Optimal Spilling for CISC Machines with Few Registers" (Appel and George, ACM Symposium on Programming Language Design and Implementation, June 2001) is an 11-page paper with a one-page appendix giving an actual program (not pseudo-code).

?? "Proof-Carrying Authentication" by Andrew W. Appel and Edward W. Felten (ACM Conference on Computer and Communications Security, 1999), a 13-page paper containing about 13 lines of Twelf code.

?? "Hints on Proving Theorems in Twelf" by Andrew W. Appel, technical report, February 2000, a 43-page tutorial that is more than 50% Twelf code.

9. But in most cases the computer program is thousands of lines long, and is best examined by its readers not in printed form, but on a computer system. Therefore, in a more typical case I put the software on the Internet separately from the paper. For example:

?? "Standard ML of New Jersey," compiler software my colleagues and I put on the Internet 1988-2001. This is a very substantial piece of software -- several hundred thousand lines of source code -- written by many researchers; and more than twenty scientific papers by these authors, individually and jointly, describe different parts of the program. By putting this software on the Internet we enable other researchers to use it as infrastructure for their projects, and to study and modify our own source code for their research.

?? "VM-PUP," a computer program benchmark related to the paper "Virtual memory primitives for user programs," by Andrew W. Appel and Kai Li (International Conference on Architectural Support for Programming Languages and Operating Systems, 1991).

?? "Zephyr ASDL," translation software related to the paper, "The Zephyr Abstract Syntax Description Language" by Daniel C. Wang, Andrew W. Appel, et al. (USENIX Conference on Domain-Specific Languages, 1997).

10. Like most computer scientists, in my own research I rely heavily on computer programs published by other scientists. The Twelf system by Frank Pfenning of Carnegie-Mellon University, the SPIM system by James Larus of University of Wisconsin, the Lambda Prolog system by Dale Miller of Penn State University, the VPO system by Jack Davidson of U. Virginia, the SUIF system by Monica Lam of Stanford, and the Edinburgh ML compiler from the University of Edinburgh, are just some examples of the academic research software that I have relied on in my own work. In each of these cases, the computer programs were distributed on the Internet to accompany scientific papers; I can study the inner workings of the programs to deepen

my understanding of the research papers, and I can use the programs as the infrastructure for building my own scientific software.

11. **Case studies in computer security research:** I have supervised several undergraduate projects in security analysis and reverse engineering of existing systems. Peter Ullman built tools for reverse engineering object-code programs, with the goal of automatically protecting host computers from viruses; Mr. Ullman is now a patent attorney. Andrew Myers defeated the authentication protocol of a networked computer game and implemented an automated player client, in the process learning about the limits to "trusted systems"; Mr. Myers is now a graduate student working on computer networking at Carnegie-Mellon University.

12. At Princeton in the autumn of 1995, one of my graduate students came to me and explained that Sun Microsystems was advertising Java as a safe platform to run untrusted programs (applets) in a Web browser, but he and another student had found six different ways to break the security, allowing "rogue applets" that do nasty things to an unsuspecting user. They were preparing a paper describing the weaknesses in Java security.

13. The students' main concern was, "Is this research?" They wanted to know if the paper would be publishable, whether they could build the beginnings of a scientific career on this kind of work.

14. My answer was, "Of course this is research." If everyone in computer security does synthesis work without any analysis of others' systems, no substantial progress can be made. As it turned out, I was right: it was publishable and of great interest to the computer science community and beyond. The two students were Drew Dean and Dan Wallach, both of whom are Plaintiffs in this action. The paper they wrote (joined by Professor Ed Felten, who helped them

develop the ideas further), "Java Security: From HotJava to Netscape and Beyond," was accepted for publication in the competitive IEEE Symposium on Security and Privacy, May 1996. This analysis research was followed by good work in synthesis that drew on the results of the analysis: Drew Dean's "The Security of Static Typing with Dynamic Linking" (ACM Conference on Computer and Communications Security, 1997) explained a solution to one of the security problems they found in that 1996 paper; Dan Wallach and Ed Felten's "Understanding Java Stack Inspection" (IEEE Symposium on Security and Privacy, 1998) explains a solution to one of the other problems.

15. **The SDMI Challenge:** I was not a part of the team of researchers who analyzed the watermarking and other technologies of the "SDMI Challenge," but I did observe behaviors by the scientists/authors that were significantly more inhibited than the computer-science norm in publishing their results:

?? Most researchers post to the Web early versions of their research papers as soon as the papers are finished, i.e. at the time of submission to a conference. (Today I found that 12 of the 23 papers to be presented at the upcoming International Conference on Functional Programming are available on their authors' web pages; see the Appendix to this declaration.) The paper "Reading Between the Lines: Lessons from the SDMI Challenge" was not so posted, even though the authors took the trouble to post an announcement that the paper existed.

?? Most academic researchers freely give pre-publication copies of their papers to colleagues at other universities who specifically request them. In the face of many such requests, Felten et al. did not distribute any such copies. Although Ed Felten showed me

a copy, he refused me permission to assign it as reading for my undergraduate class, a most unusual action in computer science.

?? Most computer scientists use fragments of program code or at least pseudo-code when it is the most effective way to illustrate the ideas being presented; the version of "Reading between the lines…" submitted to the Information Hiding Workshop contained no such fragments, even in places where it would have been helpful.

?? And, of course, computer scientists whose papers have been accepted for presentation at competitive conferences almost invariably show up and explain their results. Until this year I would have said "invariably," since in the dozens of conferences I have attended I have never seen a paper withdrawn for reasons even remotely similar to the situation here.

16. Since April 26, when the researchers withdrew their paper, many members of the Princeton faculty from many departments -- Computer Science, Electrical Engineering, Geosciences, Music, Philosophy, Physics, Sociology -- have expressed to me their outrage at the censorship of scholarly publication and their support for these researchers.

17. At the Princeton University Faculty meeting of April 30, 2001, there was a discussion of the SDMI incident. The meeting was unusually well attended, I believe because this item was on the agenda (even though inserted at the last minute, on April 27). The Faculty voted unanimously for a motion to study how Princeton University can best defend academic freedom against censorship by threats of litigation.

18. **Troublesome aspects of the DMCA:** The Digital Millennium Copyright Act is particularly troublesome for computer scientists because (a) it's not at all clear what is covered

under the term “circumvention device,” and (b) technological usage controls (such as cryptography and watermarking) prevent scientists from using automated tools for the scholarly analysis of published works.

19. The DMCA raises a number of questions that affect the work of computer scientists. Before the SDMI incident, I would never have imagined that a scientific research paper would run afoul of the DMCA. Will it be true that any discussion of a weakness of a security scheme (that could possibly be used for access or copy control for copyrighted works) will be actionable? Will it be actionable only if the discussion mentions technical details? Or only if the discussion is in writing? Is any explanation of the inner workings of an access or copy-control measure actionable, or only if it uses computer source code to illustrate the point? If computer source code is actionable, is pseudo-code permissible? What about a formal English-language explanation that could be translated into computer source code?

20. Many researchers in computer science, information science, library science, musicology, film studies, and other disciplines design and use sophisticated software tools for the scholarly analysis of published works. In February 2000, Ed Felten and I wrote a paper, “Technological Access Control Interferes with Noninfringing Scholarship,” explaining how this kind of research requires fair-use access to (digital) works in unencrypted form. My colleague Peter Ramadge, Professor of Electrical Engineering at Princeton, does research in “video content analysis”; as he testified in Universal City Studios v. Reimerdes (111 F. Supp.2d 294), he has designed software that will analyze camera angles in a digital video of a soccer game or a movie. He has been stymied by DMCA-sanctioned content protection of DVD movies. Although in principle he could negotiate a license from the copyright holder, in practice he has found it

difficult to obtain such licenses: scientists at universities are not well equipped to identify the copyright holder, find the actual person from whom to seek licensing rights, and negotiate a license, all for what is really fair use of the material anyway. He explained in his deposition and testimony (Universal City Studios v. Reimerdes) the cumbersome and restrictive arrangements that he and others use with industrial partners.

21. My colleague Perry Cook, Associate Professor of Computer Science and Music, does research in audio analysis: his software can "listen" to a radio broadcast and determine the genre of the radio station (Top 40, Classic Rock, etc.; "Automatic Musical Genre Classification of Audio Signals," by George Tzanetakis, Georg Essl, Perry Cook, submitted to International Symposium on Music Information Retrieval, 2001). If music is subject in the future to DMCA-sanctioned technological usage controls, Professor Cook might have to avoid analyzing much of the music on the Internet.

22. **The future of computer security research:** Although I started this declaration by explaining that good research needs analysis followed by synthesis, in practice many computer scientists find it all too easy to leave out the analysis. After all, analysis requires an understanding of someone else's system, whereas synthesis means designing one's own system. There is always a solipsistic temptation to ignore the world and construct self-contained, artificial, ideal system of no relevance to the real world. In computer science, analytic research is rarer than synthetic research.

23. Now imagine a world in which analytic computer security research -- which in practice often means a concrete demonstration that someone else's security system has specific weaknesses -- is subject to threats of litigation. Not only the speech of any potential researcher

will be chilled, but the entire research direction of the field will shift away from analysis. This is fundamentally the problem with the DMCA. The United States would be leaving it to overseas researchers to conduct analytic research.

I declare under penalty of perjury that the foregoing is true and correct and that this Declaration is executed in Princeton, New Jersey on August 1, 2001.

_____

Andrew W. Appel

Appendix: Pre-conference availability of papers from authors' web sites.

On June 14, 2001 I visited the Web site of the International Conference on Functional Programming (http://cristal.inria.fr/ICFP2001), whose papers will be formally presented in September 2001. I found the list of accepted papers below. I then used the Google search engine to find as many papers as I could from their authors' web sites. In each case where I found a copy of a paper I have listed the web address. Overall, 12 of the 23 papers have been posted by their authors before the conference.

1. Optimizing Pattern Matching, by Fabrice Le Fessant and Luc Maranget
2. Generic Unification via Parameterized Modules, by Tim Sheard

   http://www.cse.ogi.edu/PacSoft/publications/2001/sheard.pdf
3. Automatic Generation of Staged Geometric Predicates, by Aleksandar Nanevski, Guy Blelloch, and Robert Harper
4. Type-Based Hot Swapping of Running Modules, by Dominic Duggan
5. Generic Validation of Structural Content with Parametric Modules, by Tyng-Ruey Chuang

   http://www.iis.sinica.edu.tw/~trc/tr005.ps
6. A Simple Implementation Technique for Priority Search Queues, by Ralf Hinze

   http://www.cs.ruu.nl/people/ralf/publications/ICFP01.ps.gz
7. Events in Haskell, and How to Implement Them, by George Russell
8. A Dependently Typed Assembly Language, by Hongwei Xi and Robert Harper

   http://www.ececs.uc.edu/~hwxi/academic/papers/DTAL.pdf

9. Recursive Structures for Standard ML, by Claudio Russo
10. Developing a Stage Lighting System from Scratch, by Michael Sperber
11. Extensible Algebraic Datatypes with Defaults, by Matthias Zenger and Martin Odersky

    http://lampwww.epfl.ch/~zenger/docs/icfp01.ps.gz
12. On Regions and Linear Types, by David Walker and Kevin Watkins

    http://www.cs.cmu.edu/~dpw/papers/lr-submitted.pdf
13. Functional Array Fusion, by Manuel M. T. Chakravarty and Gabriele Keller

    http://www.cse.unsw.edu.au/~chak/papers/fastarrays.ps.gz
14. Compositional Explanation of Types and Algorithmic Debugging of Type Errors, by Olaf Chitil
15. Macros as Multi-Stage Computations: Type-Safe, Generative, Binding Macros in MacroML, by Steve Ganz, Amr Sabry, and Walid Taha

    http://www.cs.indiana.edu/hyplan/sganz/publications/icfp01/paper.pdf
16. Real-time FRP, by Zhanyong Wan, Walid Taha and Paul Hudak

    http://cs-www.cs.yale.edu/homes/taha/publications/preprints/icfp01-pre.ps
17. Cost Recurrences for DML Programs, by Bernd Grobauer

    http://www.brics.dk/~grobauer/papers/cost_dml/index.html
18. Contification using Dominators, by Matthew Fluet, Stephen Weeks

    http://www.soi.city.ac.uk/~ross/papers/notation.ps.gz
19. A New Notation for Arrows, by Ross Paterson
20. Down with Emacs Lisp: Dynamic Scope Analysis, by Matthias Neubauer, Michael Sperber

21. Functioning without Closure: Type-Safe Customized Function Representations for Standard ML, by Allyn Dimock, Ian Westmacott, Robert Muller,
22. Franklyn Turbak, J.B. Wells
23. Possibilities and Limitations of Call-by-Need Space Improvement, by Jörgen Gustavsson and David Sands

    http://www.cs.chalmers.se/~gustavss/drafts/spacedraft.ps
24. Charting Patterns on Price History, by Anand Saswat, Wei-Ngan Chin, Siau-Cheng Khoo