Summary of testimony of **Andrew W. Appel** before N.J. State Senate, State Government Committee, May 26, 2005.

My name is Andrew Appel. I am a Professor of Computer Science at Princeton University, where I have been on the faculty for 20 years. My specialties in research and teaching include computer security and the design and analysis of computer software, especially at the interface between software and computer hardware. The kind of problem I study is, how do you analyze a piece of software to tell whether it does what it's supposed to, and how do you tell whether software has security weaknesses that permit fraudulent operations?

These questions are particularly relevant to voting machines. In fact, I have studied voting machine issues in great depth over the past year, and last fall I taught a course at Princeton University on election machinery, particularly as it relates to practices in New Jersey.

I support S.29/S.2463, requiring a voter-verified, auditable paper record of each vote cast. With paper ballots, the voter gets a chance to make sure that his or her vote is recorded accurately, and in a recount, Republican and Democratic observers can see for themselves that county officials are adding them up correctly. S.29 would be an even better bill if it provided for mandatory recounts of randomly selected precincts, to make sure there's been no funny business with the machines. In addition, I would say that precinct-based optical scan technology has many advantages—not only does optical scan have a voter-verified auditable paper record, but voting doesn't have to be interrupted if the machines break down, and it's easier to accommodate unexpectedly large voter turnout.

Without voter-verified paper records, we would have to trust that the voting machine software and hardware is working correctly. As I'll explain, it's not realistically possible to verify that voting-machine software will accurately count the votes. I'll explain what the problems are, and why there are no good solutions.

When you push the button for your candidate, a computer program inside the machine decides what to do about it. It's supposed to add your vote to the total for your candidate; this total is not a mechanical counter, it's just a number in the memory of the program. It’s easy to write a program that will count the votes correctly; but it’s also easy to write a program that cheats, that moves half the Republican votes into the Democratic column. Of course, when election officials test the machine before the election they might notice that! But computer programs are very flexible; we can make one that behaves correctly on every day except the first Tuesday after the first Monday of November, and cheats just between noon and 5 p.m. on election day; or only cheats after the first 100 votes are cast; or only cheats after a certain write-in vote is entered. Voting on a direct-recording electronic machine without a voter-verified paper record is like walking into the booth and telling your choice to a little leprechaun; you have no way of knowing whether he’ll accurately remember it.

You might think that someone could look at the software in the machine to make sure it doesn’t cheat, but that’s not really possible. In my testimony today I have time to explain just two of the reasons why that doesn’t work. First, it is prohibitively difficult to analyze the computer program inside a voting machine to make sure that it accurately counts the votes under all circumstances; and second, even if the that could be done, it’s really difficult to tell whether that program that was examined and analyzed is still installed in the machine on election day, or whether it has been fraudulently replaced with another program that manipulates elections by adding votes to the wrong candidate.

First, the difficulty of analyzing computer programs. The typical voting machine has about 30,000 to 60,000 lines of computer source code. If we think of a computer program as a kind of “machine,” that’s like a machine with 30,000 different moving parts—an incredibly complex device. It’s incredibly difficult to analyze how all those parts will interact with each other in response to any conceivable combination of inputs. Computer scientists have been working for years on this, and I can tell you that while we continue to make progress on this problem in the laboratory, the problem is far from solved for real-world applications like voting-machine software.

Consider this: major software vendors, who have every incentive to produce good programs, still can’t manage to produce software that’s perfectly reliable and resistant to fraudulent takeover by computer viruses. For example, if your computer uses the Windows operating system, every week or two you’re asked to update with new patches to correct bugs and security vulnerabilities.   Microsoft spends a billion dollars a year in software testing and software review, and still there are bugs, and still there are security holes found by malicious outsiders every month: that's the state of the art. For any computer program of substantial size, it's impossible for even the most expert computer scientists to guarantee its correct operation without an expenditure of millions of dollars to pay for hundreds of person-months of effort.

The problem is actually worse than that with voting machines. For most kinds of software we can assume that nobody inside the company would have much to gain by making the program malfunction. But in elections, the stakes are very high: there's certainly enough motive for an insider at a voting-machine company to try to throw elections by writing fraudulent software. It's easy to write software that behaves well whenever it's tested—on every day except the first Tuesday after the first Monday in November—but moves votes around during the election. Furthermore, an election-fraudster could try to hide his fraud among the 30,000 or 60,000 lines of legitimate software, making it even more difficult to detect. Even without a fraudulent insider, this could happen; for example, when computer viruses take over your machines, it's not because there's a fraudulent insider at Microsoft corporation, it's because even Microsoft finds it very difficult to write software without vulnerabilities that allow fraudulent outsiders to take control. In summary, we cannot rely on the computer software in a voting machine, and we cannot effectively review and audit that software.

Finally, even supposing that New Jersey were willing to spend the millions of dollars that it would take just to audit and review the computer program designed by a voting-machine manufacturer; we have no way of knowing whether that program is the one actually installed on the voting machine at election day! If we ask the machine to print out what program is installed, then we're really asking the software in the machine to tell us about itself. It's easy to write software that cheats on the election, but when asked about itself, will print out a copy of the certified software.

On some models of voting machine, such as the Sequoia AVC Edge, used in one county in New Jersey, installing new software is as easy as inserting a smart card and typing in a password. My colleague Professor David Dill, a computer scientist at Stanford University, had this procedure demonstrated to him by a county election official in Santa Clara, California. On other models, such as the Sequoia AVC Advantage used in Mercer County, a simple hardware modification is necessary, as I'll demonstrate right now with this circuit board.

Inside a voting machine is a circuit board about a foot square, depending on what model of machine it is. I have here a similar circuit board, from a personal computer but very similar in technology to what voting machines use, and you can see here two important components: the central processor that executes the instructions and the ROM memory that contains the instructions. These instructions control everything the computer does; you could put instructions here for the Space Invaders video game, or for a voting machine, or even for a voting machine that fraudulently moves votes from the Republican column to the Democratic column. Now suppose I show up at an elementary school or a firehouse in Mercer County the day before an election or the day after. I'll find a few Sequoia AVC Advantage machines, unattended—most schools don't have round-the-clock security guards. I've seen the machines just sitting there in the John Witherspoon Middle School lobby two days after the recent school board election. I could pick the lock on the cabinet, unscrew 10 screws to remove a metal panel inside, and pull out the software ROM just like this. Now I can install a fraudulent ROM that cheats on elections, like this. Screw the screws back in, and now this machine will cheat on elections for the next 20 years.

The only foolproof way to check that the computer still has the authorized software is to pull out this ROM chip—let me do that now—and install it into an analyzer machine to examine its contents. Are we prepared to let each of the partisan pollwatchers do that on the morning of election day? That doesn't seem like a good idea. But it illustrates the lengths we'd have to go to, if we try to audit elections without a voter-verified paper record that permits recounts that are independent of the computer software.

If the machine prints out voter-verified paper ballots, so that we can conduct recounts to check that the machine is accurately counting votes, then erroneous or fraudulent software will be caught. Without a voter-verified paper record, there's no way to be sure that the vote totals reported by the machines actually correspond to the voters' choices. I urge you to provide for voter-verified paper ballots, and to remember that this can be accomplished not only by equipping DRE machines with printers, but perhaps even more effectively by the use of optical-scan ballots.