David Jefferson Updates The following document is alternating black and blue text. The blue is mine; the black is a widely-circulated document from the Election Center that attempts to defend the security and reliability of DRE voting systems, and implicitly argues against any need for them to be upgraded to include a voter verification feature. Since the original document contains a very large amount of misleading or completely incorrect information, I feel that it is necessary to circulate this critiqued version.

Unfortunately, the arguments contained in the original show considerable misunderstanding of the software development and security issues for DREs, and of the range of security and failure threats inherent in them. Without presenting any realistic threat model, the author argues that certain attacks he outlines are unlikely to be successful; but he clearly does not understand the technical issues, and takes no account of much simpler and more dangerous attacks than those he discusses. I don't have space to present the full range of security vulnerabilities in DRE systems without voter verification, so I confine myself to refuting the arguments included here. However, if you would like further information, feel free to contact me by email.

Addition in 2005: It has come to my attention that this Election Center document is still circulating and being republished over two years after I refuted it. Frankly, I think the document should have been withdrawn long ago, but it persists. In this current version I have updated my arguments to reflect the current state of the debate over election security; but the general tone and conclusions of this rebuttal are the same.

David Jefferson
Lawrence Livermore National Laboratory

Chair, CA Secretary of State's Voting Systems Technical Assessment and Advisory Board
CA Electronic Voting Task Force Member
CA Internet Voting Task Force Technical Committee Chair
National Science Foundation Panel on Internet Voting
d_jefferson@yahoo.com

The Election Center an international association of voter registration and election officials
12543 Westella, Suite 100 Houston, TX 77077 Phone: 281-293-0101 Fax: 281-293-0453

Email: electioncent@pdq.net Website: www.electioncenter.org

Now that Direct Recording Equipment (DRE) voting systems are growing in acceptance and use in American elections, it is almost inevitable that some groups, individuals and organizations will claim that such systems are not safe enough to use in elections.

People are entitled to doubt the security and reliability of voting machines. The burden of proof that voting systems, including DREs, are accurate, reliable, and secure rests squarely on the voting system vendors. If they are unwilling or unable to provide that proof, then the public is entitled to—and in fact must—reject the systems in question, or demand improvement.

And this argument is not new. When lever machines were first introduced into the elections process, all those favored paper used the same kinds of arguments. When IBM first started computer counted punch card voting, many of the same kinds of arguments were made.

And when new technologies are introduced and experts point out security or reliability problems with them, it is also not new for many people to argue that those security problems are exaggerated: that the attacks hypothesized are extremely unlikely, or would not succeed, or would be detected early, or could not happen because no one has the access, resources, knowledge, or motivation to conduct them. Often those arguments are naïve, ill-informed, and based on limited understanding of software or security. When remote Internet voting was first introduced in the U.S. in 2000 (and attempted again in 2004), many of those same arguments were made. Fortunately, with enough time and opportunity to present the case, the expert opinion that PC-based Internet voting introduces potentially catastrophic security vulnerabilities was finally accepted, and possible disaster was averted (for the time being). Most of those same experts are also concerned about DRE security.

Because DRE's represent another shift in the kinds of technology used for elections, we see the renewed fears of introducing the newer technology. It is entirely normal for these arguments to arise as we shift to a generational change in the types of voting systems used.

I do not argue that DREs are in principle a bad idea; indeed they have *some* real advantages to ordinary voters, to the disabled, and to voters prefer read a language other than English. But as currently designed, paperless DREs have fatal security and privacy flaws so dangerous that they could allow people with access to the software to modify election results on a national level, and *without detection*.

It is a matter of national security that we fix these flaws. Fortunately most of these flaws can be fixed with a single feature, voter verification, which simply allows voters to verify that their votes are cast as intended, and at a time in the voting transaction when the vote cannot be secretly overwritten by software without detection. What we argue for is not the elimination of DREs, but the immediate requirement that they be augmented with voter verification technology.

Many of us arguing for voter verification are computer scientists, and hardly harbor any general "fears of introducing ... newer technology". We spend our professional lives creating

new technology and helping to introduce it. So when the computer science community, usually technology boosters, is nearly unanimous—which it is—in warning that DREs without voter verification have huge and glaring security vulnerabilities, you can be sure it is not based on simple fear.

The problem is that well intentioned people, some of them even highly educated and respected, scare voters and public officials with claims that the voting equipment and/or its software can be manipulated to change the outcome of elections. And, the claim is, it can do so without anyone discovering the theft of votes. Since so many people tend to distrust technology they have limited knowledge about, it only makes the situation worse.

Yes, I do claim that voting software can be manipulated to change the outcome of elections, and quite possibly without detection. No one is specifically trying to scare anyone. But when we carefully explain the dangers of introducing DREs without voter verification, the implications are indeed scary. It is vital that security problems with voting systems not be covered up in the name of voter confidence.

Let's confront the problem directly: it is highly probable that any machine devised by humans can be broken by humans. So ANY technological argument to the contrary seems to be doomed from the very beginning. We can take precautions, we can make it more difficult, but the end analysis is that you cannot build a totally secure voting device.

No one argues that with voter verification DREs become "totally secure voting devices". But one has to recognize that some bugs and attack threats are much more serious than others, and we are concerned about the most serious ones.

The most dangerous potential vulnerabilities (1) can affect hundreds of elections simultaneously, rather than just one; (2) are easily hidden and unlikely to be detected; (3) can be perpetrated by a single person without requiring a larger conspiracy; or (4) are particularly easy to perpetrate. DRE software, and the associated development and distribution processes, should be designed so that attacks like these are virtually impossible; but sadly, they are not. Fortunately, most of the serious bug and attack scenarios can be prevented or at least detected and ameliorated with the addition of one mechanism, voter verification, which is why we advocate it.

The real question is, can you gain access to the software, change it, have it manipulate the results for one or more races, have it not be evident when you do the pre-election test, erase itself before the post election test, and get away with it totally undetected?

The short answer is simply “Yes”. However, as this question is phrased it shows a serious misunderstanding: it is not necessary for malicious software to “erase itself before the post election test”, since it is no harder to pass undetected through the post election tests than to pass all of the other tests. All that is required is for the malicious software to go silent; it need not erase itself.

As usual, this all boils down to appropriate election policies and procedures, and with an understanding of what it would take to do all of the above and get away with it without anyone discovering what you did (or attempted).

There are presumptions built-in to this paragraph that do not apply to the most damaging attacks I have in mind, namely vendor insider attacks. With those attacks, by the time the “election policies and procedures” are invoked, it is already *way too late* to do anything about them, possibly by several years. It is important to understand that no policies or procedures can repair votes that are recorded incorrectly by DRE software in the first place. Moreover, the vendors and testing authorities have maintained a total veil of secrecy over these systems, so no one knows what other vulnerabilities may exist, or what procedures would be appropriate to minimize them.

Voters need to know that even if the election official is sloppy about some procedures, that it is still improbable (vs. impossible) a "rogue vendor" could act alone to change election results (to use an allegation that has been made).

This argument is wrong in several ways.

First, it is completely inappropriate to apply probability arguments in matters of human behavior. It is a common error to confuse *reliability* issues (in which no human attack is involved and in which probabilistic and statistical arguments are central) and *security* issues, (in which intelligent human attackers are involved and in which game-theoretic treatment rather than probability is more appropriate). In security issues, one simply cannot estimate such probabilities, because no one understands the vast complexities of human motivation. On September 10, 2001 most people would have judged that it was highly improbable that the next day four airliners would be hijacked and crashed into famous landmarks. But they would have been wrong.

Second, if you do make a probabilistic argument you have to be quantitative; it is of no value to just assert that an event is “improbable.” People make snap judgments about probability all the time, based on no analysis at all, that can be off by many orders of magnitude. There is a 10,000-fold difference between a probability of 1/100 and 1/1000000, yet each would be called “improbable”.

Third, even if we could estimate the probabilities in a meaningful way, it is not the probability *per se* of an event that matters, but the *risk* it entails, which is calculated by multiplying the probability of an event by the damage that results when the event happens. In the case of a vendor insider attack on a DRE system, while we cannot estimate the probability of an attack, we *can* estimate the amount of damage that could be done. Almost certainly 1% or 2%, or in some cases more, of all votes cast nationally on a particular vendor's equipment could be switched *without detection*, changing the outcome of many races across the country, including even the presidency in a close election such as we had in 1960 or 2000.

Here are the steps that a person would have to go through to be able to change the outcome of an election.

There seems to be unwarranted presupposition here that the goal of an attack would be to rig the outcome of some *particular* election, e.g. the California gubernatorial race. A much more powerful attack, and one more suited to the vulnerabilities in DRE technology, would be to change the outcome of a large fraction of *all* very close elections in the nation, without targeting any particular one. Such a goal is much easier to achieve with current DRE vulnerabilities, and much harder to detect. And, I must point out that we don't know what other vulnerabilities may be lurking in these systems because, as will be discussed later, the code is secret. There may be very effective attacks that we haven't thought of.

Throughout the following points, there are numerous presuppositions about the kind of attack that the writer imagines, presuppositions that I will try to point out as we go along. Unfortunately author does not acknowledge, or respond to, more serious attack scenarios.

A) You have to know the language the software was written in (not English, Spanish, etc., but rather the programming language)

In one sense this is true. The attacks we are most worried about are by programmers, or at least people with strong IT skills. But it is not true that you have to know in advance what language a program is written in in order to modify it or attack it. And in any case, an experienced programmer knows lots of languages, and programming skill is largely language independent.

B) You have to know every location in the software where it checks on itself to verify that the numbers it is reporting are accurate.

This is totally incorrect in a couple of different ways. There is no way for a DRE to check that what it reports is "accurate", in the sense of correctly capturing the voter's intent. Such a test can only be done by the voter comparing his or her intent to the output of the DRE, and indeed, that is the whole point of the *voter verification* requirement for DREs.

Instead of checking "accuracy", the best that can be done inside a DRE is to check that the software encounters no logical inconsistencies. There should, of course, be many internal consistency checks in DRE software—that is always good engineering practice. But if a vote is surreptitiously switched from one party to another in the *original* vote copy before any duplicates are made, or if the voter is tricked by the software into "voting" for the wrong candidate, then no consistency check can possibly discover the problem because *there is no inconsistency,* i.e. the voter *could* have intended to vote that way.

C) You have to know the language AND VERSION of the compiler that was used to compile the program (it converts the program from a human readable form to machine language)... in order to "reverse engineer" the software you must have the identical version of the compiler in order to reverse engineer it;

This paragraph is completely wrong. In the first place, it presupposes a reverse engineering attack, but there are plenty of attacks that do not depend on reverse engineering, such as an insider attack where the perpetrator already has easy access to source code. Second, one does not have to know anything about the compiler used in order to do reverse engineering. Third, if for some reason you do want to know the exact compiler and version used, it is not hard to figure out from the object code; in most cases there are only a handful of possible choices. Furthermore, no compiler I know of even tries to disguise this information; sometimes the compiler actually includes it directly in the object module (for security reasons)!

D) You have to gain access to the software for a long enough period to actually replace it;

Vendor-based insider attacks do not require "gaining" access, since the access is always available to the owners of the software. (There are also several other ways to get access to the software if the attack is not from inside.) And since there are generally months between consecutive elections, there would seem to be plenty of time to "actually replace it" if that is required. But it is also important to note that not all attacks even require replacing the software running on DREs; some require only exploiting a vulnerability in it.

E) You have to make the software ignore the pre-election test or tests and only initiate itself on election day;

Contrary to the intuition of people not skilled in security analysis, this is very easy to do. The basic technique is for the software to detect whether it is running as part of a test, as opposed to a running in a real election, and for it always to behave honestly in any test.

The only kind of pre- or post-election test that would have any hope of detecting a well-designed fraud embedded in DRE code would be a test that looked *exactly* like a real election to the software, i.e. a test that takes about 13 hours and has the right kind of vote timing and

frequency of human errors, the right range and distribution of votes, and had all other details exactly right. Even with such a carefully-designed test, fraud still might not be detected because cheating could be randomized so that it is not repeatable, and is therefore difficult to distinguish from a timing bug or operator error.

F) You have to have the software be able to actually change votes throughout the day and do so undetected;

This is wrong. The DRE software might, for example, do all of the vote switching in a few milliseconds during the shutdown procedure at the end of the day—which is actually the optimal time, I believe. But it is also perfectly possible to change votes all day during the voting process, and there are a hundred ways to do so. These vote changes will not be detected by election official because *no one is allowed to look at vote subtotals during an election*; doing so would compromise voter privacy.

G) The software must be able to erase or conceal itself before any post-election test.

The software does not have to "erase itself" before any post-election test. All it has to do to "conceal itself" is not cheat during any post-election test, so that it will not be detected. And that is just a special case of the more general rule that, in order to avoid detection, malicious software should not cheat during any kind of test designed to look for software problems.

H) If the software is programmed onto a ROM (Read Only Memory) chip then you have to have physical access to the units.

This also is wrong in several ways. In the first place, all DREs are subject to software upgrades and patches, so nowadays the software is not in ROM at all. Instead they use Flash (or EEPROM) memory to hold the software and the votes. This is a vital distinction because unlike ROM, these latter types of memory are completely writable under software control, making error and fraud vastly easier.

Secondly, one does not necessarily need physical access to the software chips in order to attack them. By far the easiest mode of attack is to tamper with the software while it is in development, or during upgrades, bug fixes, or other software modifications, so that the tainted code is loaded onto the chips through the regular distribution channels. With that kind of attack, one never has to go near the voting machines, or ever set foot in the counties that use them, in order to swing elections.

The next three points presuppose that an attack requires physical access to voting machines while they are in the county's custody. In my opinion, such attacks are not the most "attractive" ones; they would be too complex and not likely to succeed, and in any case would affect votes only in one county. The easier and more dangerous vendor insider attacks are simply not discussed here.

I) With access to the units, you must be able to remove enough of the ROMs in the units to reprogram them. This entails having enough time to either erase the ROMs installed in the units or having enough supplies of identical ROMs that you can have them preprogrammed and inserted into the units... all undetected.

As I said, the easiest attacks do not require physical access at all.

J) You then have to have access a second time to remove the "malignant" ROMs after the election and replace them with the real ones you removed (so that you can get away with the election fraud undetected).

Again, the easiest attacks do not require physical access at all.

But in any case, if an attacker has gone to the trouble to insert malignant "ROM's" into voting machines (remember, it is Flash memory, not ROM), why would s/he need to remove it? Chips with malignant code look just like the originals, so no one will detect the replacement. It may be much better from the attacker's point of view to leave the tainted chips in the DRE and so they can continue to do their dirty work in future elections.

K) You have to do this not only on enough machines in one jurisdiction (unless your intent is to manipulate a local election – and why would anyone take these kinds of risks for a County Commissioner's race, or Sheriff's race or Mayor's race?), but in many jurisdictions in order to steal a Congressional race or state race? And for the presidency, this would involve thousands and thousands of people...unless of course we go to one system nationally (or Internet voting).

I have already mentioned that the supposition that you need physical access to the machine to corrupt it is incorrect. And it also absolutely does *not* require a conspiracy of "thousands" of people to affect hundreds of jurisdictions nationally. All it requires is for one type of DRE to be used in hundreds of jurisdictions, and a *single corrupt programmer in a position to edit the codebase* for that type of DRE. His or her malicious code will be transported and loaded into many thousands of machines through routine code distribution and loading processes.

Another unwarranted presupposition is still present in this argument: that the intent of the attack is to target a particular race. As I said before, that is not the most dangerous attack goal. It is easier and more insidious just to attack all races simultaneously, and just win many of the close ones (including, perhaps, a statewide slate of presidential electors). With

vendor insider attacks, it is actually easier to attack national races than statewide races, and easier to attack statewide races than local ones.

L) In states with multiple vendors of DRE's it means that you have to go through the entire process for EACH type of DRE (and still be able to get away with it).

Absolutely wrong. Today one vendor controls over 50% of the U.S. voting system market, and there is no reason to suppose that this pattern will not continue. Now that DREs are widespread, an attack on that one vendor's system could have national effect, swinging many close races around the country. A successful attack on any one kind of widely-used machine has to be considered the electoral equivalent of a weapon of mass destruction.

M) Even in Central processing of election results through an Election Management Software package, you still have the individual results of local precincts (and each unit therein) and can verify the results as reported by them in comparison with the Election Management System results.

This is totally misleading. All that can be done with election management software is consistency checking. But *if the votes were recorded incorrectly in the first place*, as they would be in the attacks I am most concerned about, then *there is no way at all for election management software to even detect the problem, let alone fix it*.

N) In many states, there is a requirement to escrow the software, so that you can compare the software in the units with the escrowed software.

With a vendor insider attack the two code versions—the one running on the DRE and the one escrowed--will be identical, with any hidden malicious logic present in both. It may also be that the executable code is modified without modifying the source. Unless a clean build environment can be re-constructed, there is no way to check that the executable code corresponds to the escrowed source code.

In other attacks, however, a key point could be to subvert the test of whether the two code versions are the same. In some cases I have heard vendors describe how they use only a CRC (cyclic redundancy check) for this purpose instead of a cryptographic hash, in which case subversion of the test is totally trivial.

It is also important to note that it is almost impossible for anyone—even election officials--to actually "compare the software in the units with the escrowed software". Most states are bound by a contract stipulating that absolutely no one has that right except in extremely limited circumstances, such as an election where a court suspects that software problems may be responsible for erroneous results, or in case of the bankruptcy of the vendor.

O) Even in states where this is not so, NASED requires the ITAs to escrow the software at the ITA (Independent Test Authority) so it can be compared to the originally qualified software.

Same answer as for N)

P) You now have to have the involvement not just of one or two people but significant numbers of folks to make all this happen undetected, actually change the outcome, and get someone elected who should not have been elected.

No, this is incorrect. Changing the outcome of elections can be accomplished, and in my opinion is best accomplished, by a single closed-mouth criminal programmer at a single vendor. But if one considers attacks in which as few as two people conspire, then in fact an enormously larger array of very targeted attacks is possible. Large conspiracies are completely unnecessary and counterproductive.

Q) A piece of paper that the voter sees does not guarantee that the same results will be recorded within the machine – if you want to manipulate the election, show the voter whatever the voter wants to see and still manipulate it later. Security experts will still argue the value of having paper for recounts.

The whole point of the argument by critics is that the DRE *can* show the voter one thing on the screen and yet record something entirely different in the vote memory module used in the canvass, either through error or fraud. There is no effective way to prevent that in advance; but with voter verification honest election officials could detect the discrepancy, measure its magnitude, and possibly fix the count.

R) The current solutions presented by the vendors as a result of their concerns for the validity of the results have their own limitations, because:
   a. They add a printer, which can run of ink, ribbon, or paper
   b. Paper can jam
   c. Printer can be disconnected from power source
   (All of these mean having to repair the units during an election with repetitive jams, running out of paper, or ink, etc).
   d. They add weight to the units (complicating precinct setup, shifting control of delivery and setup from poll workers to expensive delivery services along

with quality control and security efforts over those services).

e. Voters can, and probably will, walk off with ballots with some of the solutions presented(vote buying?)

f. Inability of blind voters to check their ballots (Braille printing only covers 10 percent of the blind).

g. They add significant cost and complexity to the voting unit and to the skills required to support them in a voting location.

All of the above problems with printers (except (e)) have a degree of truth to them. One response would be to develop a paperless vote verification technology, rather than throwing out the verification concept in a misguided attempt to avoid paper at all cost. But even if paper ballots were the only way to provide voter verification with DREs, as they are today, then in my opinion we should do what is necessary in order to close the otherwise potentially catastrophic software security holes. Election officials have 100 years experience with managing paper ballots, so the logistics of paper management are hardly insurmountable.

h. While the voting system may accurately reflect how the voter has voted and print an accurate reflection of that vote as a receipt, what happens when the voter has electronically "cast" the ballot but now claims the printed receipt is different? You now introduce serious credibility claims that can irreparably damage the elections process...and they will insist on keeping the printed ballot as evidence of "fraudulent programming of the machines."

This argument seems also to be a strawman. Of course the voting protocol must specify that the voter can *only* protest that the paper copy is wrong *before* he or she commits the vote. After that point it must be impossible to even find the voter's ballot among the other randomized ballots, let alone entertain a protest.

Anyone who thinks that the claim of "fraudulent programming of the machines" might get some traction with the public when voters can verify their own ballots should try to imagine how easy it would be to sustain that claim when voters *cannot* do so. There are large, disaffected segments of the population that already believe that voting is rigged, and that electronic voting provides just a more efficient way of rigging them. Voter verification would greatly diminish that fear.

i. Or, once printed receipts leave the polling site (which will be difficult to prevent at the precinct level) do you now introduce the ability of

> fraudulent reproduction of printed receipts intended to confuse and contrive the process?

It is not difficult to prevent voters from walking away with the voter verified paper ballot images (which are *not* "receipts"). With current DRE designs, the voter cannot touch the VVPT at all. No one supports human-readable ballots images leaving the polling place with voters. That would open the door to vote selling and coercion.

> The point is simply this: do not be misled into believing that elections are reliant upon technology which can be manipulated. The real question of whether there "are sufficient and proper safeguards to make it highly improbable?" And the answer to that is yes. It may be possible to do many things, but like time travel (which is theoretically possible), it is highly unlikely at this time.

The point is that the software can be "manipulated" by any of the original developers or their system administrators, their managers, or even their quality assurance team. As of 2003 there did not appear to be any *effective* safeguards in place against these possibilities, in spite of vendor protestations to the contrary. The California Touchscreen Voting Task Force was, however, presented with several totally *ineffective* "safeguards" that fool naïve people who do not understand software development or security.

> Each of the systems is programmed at the LOCAL level. It is true that each local election is using the same base machine operating system but it is individually programmed locally. Manipulation of races for national or statewide offices or regional offices (Congress, state legislature) is far more difficult because it is highly unlikely that each of those races will appear in the IDENTICAL byte spot on each machine and would vary from one local jurisdiction to the next.

This is a confused argument. First of all, not only is the base operating system the same in all of the machines in the nation of a given type, but so is all of the application software. All of that software is qualified once at the ITA level, and then (supposedly) presented without change to all of the states for certification.

DRE systems are not "programmed" at the local level in the usual software sense of the word "program". It would be less confusing to say that they are "configured" with a small database that contains a description of the election to be held, i.e. the offices, candidates, propositions, voting options, etc. The actual software code of the machine is entirely written by the vendor, and *is never changed at the local level* (or, at least, never *should* be).

It is completely irrelevant that the races will not "appear in the IDENTICAL [location] on each machine". For some reason I keep hearing this strange argument repeated. Only

someone who does not understand programming at all could think that this is in any way an issue. Programmers have no trouble writing code—5 to 10 lines at most—to search the DRE's tables of candidates and offices to find particular party names, or to determine which candidates belong to which political parties, etc. The order in which candidates appear on screen is completely irrelevant to this attack, and even if it weren't, that order is determined by the very same DRE same software, so it is corruptible as well. It is hard to understand how anyone with technical knowledge could be confused about this.

Although many election officials seem unaware of it, vendor programmers have vastly more control over DRE behavior than any election official does. Election officials do not know, and cannot know, what really goes on inside a DRE. Like voters, they have no choice but to simply trust the software.

Another allegation made by some is that the software should be in the public domain rather than proprietary, leaving the impression that the software is secretly controlled by a company or individual. Simply because the software is not open to every hacker in the world, does not mean the software is not reviewed and exposed to public scrutiny.

The software in a DRE is definitely "controlled by (one) company" and most certainly is "secret". Calling it "proprietary" is just a way of saying "secret for commercial reasons". Furthermore, although the code is tested, it is barely "reviewed" at all in the sense of a serious external code review. Aside from the developers, only *very tiny* number of other people in the United States are ever allowed to see DRE software, primarily a few employees at so-called *independent testing authorities (ITAs)* like Wyle Labs in Huntsville, Alabama. And ITAs do not do anything resembling a real security review of DRE software.

Not only is the software secret, but so are all of the design documents, and even the reports of testing by the ITAs. Most of the states that certify DRE systems do so without their certifiers ever seeing a copy of the source code. What they get is a *certificate* that the system has passed ITA review and that the code is escrowed somewhere. The vendors' contracts with states specify *very* rare circumstances under which that code can be examined.

In no case is the "public", even an expert with a "need to know" such as myself serving the California Secretary of State, permitted to examine DRE code. If we measure secrecy by how few people are allowed to share in the secret, this makes DRE software more secret than most U.S. strategic military plans. And it is certainly more secret than, say, Microsoft's proprietary Windows operating system code, which is shared under nondisclosure agreement (NDA) with numerous large customers.

The national testing program for the National Association of State Election Directors (NASED) requires that the manufacturer's software must be escrowed with its written source code. The difference here is that the source code is

NOT secret. It is simply unavailable to the general public -- and that is a significant difference. There are many technologically advanced people who would love to have the opportunity to examine all kinds of software (not just that used in voting) but it is not within their purview to be able to do so. Should we open all of the software available simply because they are interested?

As I said in the last answer, the source code certainly *is* secret. It is not only "unavailable to the general public", it is unavailable almost everyone, including most people with a professional need to know, e.g. election officials, standards writers, state certifiers, and the security experts they engage to study and improve election systems. In my view that substantiates the description that the code is properly described as "secret".

The author seems to be arguing that someone who advocates that voting system software be open must argue that *all* software should be open. But voting system software is special. It serves a *public purpose*, and *only a public purpose*—and in a democracy it is *the most fundamental of public purposes*. There is nothing inconsistent about arguing that DRE software should be public while commercial software is not.

Since the source code is escrowed with our national Independent Testing Authorities, and additionally as a condition of approval in many states or local jurisdictions, it is not secret code. In an appropriate governmental investigation or court inquiry, it can be compared from the machine to the escrowed versions. This is an appropriate safeguard for the public interest.

DRE software is escrowed under provisions that in almost all states prevents even the highest officials from getting a copy of the code for examination. Escrowing is not a step toward openness, but a means of enforcing secrecy. For some unfathomable reason states have generally accepted this deal, probably because they do not have a deep grasp of software, and have been willing to both certify and use DRE systems without ever seeing the code.

Additionally, the nation's ITAs REQUIRE that they witness the build of the software so they can assure an added layer of precaution is built into software security.

"Witnessing" the final build is completely meaningless from the point of view of security, since the build process is childishly easy to manipulate without detection by a mere witness. If instead the ITA actually performed the build itself (rather than watched the vendor do so), in its own clean environment, and if the resulting binaries were then fed to the states for distribution to the counties, that procedure would provide some security by eliminating some—but by no means all—opportunities for tampering. However, so far the ITAs do not

perform their own builds, and just "witnessing" the build offers no security leverage whatsoever.

Of course with voter verification in place, it would not matter much what the software building and distribution process is; we could have high confidence that the votes would be captured honestly in spite of the code being poorly-engineered, bug-ridden, or tampered-with, because the voters themselves are encouraged to check each vote *after the DRE software is finished*.

The genius of the American democratic process is its diversity. Since we use so many different types of voting equipment, provided by so many different vendors, and because elections are controlled by so many local elections offices, it makes manipulating an election in America very difficult.

The fact that there are many vendors is completely misleading, because most of them have only a tiny share of the market. The voting system market in the U.S. is in fact highly concentrated. Once DREs are widespread, and assuming the same vendor market shares are preserved, then an attack on any one of the "big three" vendors, ES&S, Diebold, and Sequoia, can be an attack on hundreds of jurisdictions across the U.S. at the same time.

The ability to manipulate an election with DREs, combined with election practices and procedures, means it is highly unlikely to be able to do this and get away with it. You can still manipulate an election easiest with hand-counted paper ballots.

There are indeed many ways to manipulate paper elections, but the scope of such an attack is limited to one precinct or one county (except possibly for fraud conducted at ballot printing companies). The danger of a software attack on electronic voting is that, while it takes a little more skill (but nothing extraordinary,) it can affect hundreds of counties across the nation simultaneously.

The reality of a discussion on the technological possibilities of manipulation is that no one is ever satisfied with the technological arguments or counterpoints. For every technological challenge there is a technological solution or counter challenge, none of which ever satisfy the other parties.

It is not true that "no one is ever satisfied with the technological arguments"; with technical arguments there often is one side that is clearly correct. In this case, there happens to be virtual unanimity in the computer security community. We are "satisfied" that DREs without voter verification are easy targets for fraud, and that such fraud could easily be hidden so that it would be extremely unlikely to be detected by any of the methods that uninformed people

seem to have faith in, i.e. functional testing, code inspection, or comparison with expected election results.

It has to be proven that such challenges can be carried out successfully without possibility of detection when you combine the technological aspects with established testing procedures, election management procedures and public scrutiny of elections.

This is a dangerous and irresponsible argument. It amounts to an attempt to shift the burden of proof, so that it is not up to the vendors to prove that the machines are safe, but rather it is up to the critics to prove they are unsafe. It is also like saying we should not bother to try to prevent nuclear weapons from being smuggled into the U.S until it is "successfully proven" it can be done. Do we really need to have another election catastrophe (which we would be lucky to even detect) in order for decision-makers to take this problem seriously?

I cannot help but also note that the same people who argue that DRE machines are secure against attack also defend the complete secrecy of DRE design documents and code. That position makes it impossible, without committing a felony, to give a live demonstration that attacks on DRE software will work and will be virtually undetectable.

We appreciate and respect those who question the process and we understand their fears. And we do not take their concerns lightly. While conducting elections is likely to be an imperfect process, it is a process built upon more than 200 years of experience in how to provide appropriate safeguards. Like most situations in the electoral process, it rarely boils down to a technological issue. It almost always comes down to policies, procedures and people doing what they are supposed to do.

While it may be "rare" that electoral problems "boil down to a technological issue", this is one of those times. When the computer scientists and security experts who have examined the issues are nearly unanimous (which they are) in warning of the security danger of DREs that do not include a voter verification feature, I would hope the election officials and vendor groups would try to help solve the problem instead of denying its reality.