Nedap/Groenendaal ES3B voting computer

a security analysis

90% of the of the votes in The Netherlands are cast on the Nedap/ Groenendaal ES3B voting computer. With very minor modifications, the same computer is also being used in parts of Germany and France. Use of this machine in Ireland is currently on hold after significant doubts were raised concerning its suitability for elections.

This paper details how we installed new software in Nedap ES3B voting computers. It details how anyone, when given brief access to the devices at any time before the election, can gain complete and virtually undetectable control over the election results. It also shows how radio emanations from an unmodified ES3B can be received at several meters distance and be used to tell who votes what.

Rop Gonggrijp, Willem-Jan Hengeveld,
Andreas Bogk, Dirk Engling, Hannes Mehnert, Frank Rieger, Pascal Scheffers, Barry Wels

Stichting "Wij vertrouwen stemcomputers niet"
(The “We do not trust voting computers” foundation)

`info@wijvertrouwenstemcomputersniet.nl`

Last revision: October 6, 2006 20:59

`http://wijvertrouwenstemcomputersniet.nl/Nedap-en`

Abstract

The Nedap ES3B electronic voting computer is a system that belongs to the DRE (Direct Recording Electronic) class of voting computers. As such it only records the votes in memory. The system requires ultimate trust, since it produces an official election outcome that cannot be verified independently. In this paper we describe the results of an independent review of the Nedap ES3B electronic voting computer that was done without consent of the manufacturer, without access to source code, and within roughly one month. This paper details all the steps we needed to take to create and install our own demonstration software on the machine, as well as a modified version of its own software: a version that lies about the election results. It also details a practical attack that allows a remote observer to get some information about what is being voted on an unmodified Nedap ES3B computer by exploiting compromising radio emanations from the device. In this paper we show that the over-all security design of this computer relies almost solely on the near-universally deprecated concept of ‘security by obscurity’. Since the problems we found stem from the very design philosophy, we see no quick fixes that could make this device sufficiently secure.

We conclude that the Nedap ES3B is unsuitable for use in elections, that the Dutch regulatory framework surrounding e-Voting currently insufficiently addresses security and we pose that not enough thought has been given to the trust relationships and verifiability issues inherent to DRE class voting machines.

1. Preface

We, the authors of this paper, are part of a growing group of computer experts that opposes the use of electronic voting technology that is built in such a way that the outcome of an election is not voter-verifiable. We believe public elections are pointless unless people have the right and the meaningful possibility to verify that that their votes are counted correctly. We further strongly believe that trade secrets, secret computer programs and secret test reports have absolutely no place in any democratic election.

Given the fact that the technical specifications and source code to most e-Voting systems are not publicly available, much of what the world knows about the technical inner workings of these closed systems comes from reports such as this one or the recent work done by Princeton University researchers (Feldman et al., 2006). It is a sad fact that public awareness of something as basic as the inner workings of the modern ballot box depends on reports written by researchers that managed to somehow get their hands on a well-guarded piece of *secret voting technology* so they can take it apart. We see grave danger looming if our children grow up thinking it's quite normal for them not to be allowed to know how elections work. We sincerely hope this report will help convince the reader to do his or her part to make sure 'black box voting' is abolished.

This report openly discusses vulnerabilities that affect the particular voting system used by 90% of the voting public in The Netherlands. Any vulnerabilities discussed herein affect the very foundations of our democracy. Some will argue that by discussing and demonstrating these vulnerabilities, we are helping the bad guys. Some might even argue we created a problem that did not exist before. This is the full disclosure debate, and although it predates the invention of computers, it has been a lively part of the computer security community culture for decades. It's the debate between those that feel vulnerabilities should be told only to the manufacturer and people that feel all those affected by a vulnerability have a right to know so they can decide for themselves how much trust they place in a given system. In the case of a voting system, it is obvious that any security problems could affect all of society.

We took apart these computers because we believe we had a right to find out how our own elections work. We published this paper because we believe that you, the voter, have a right to know what independent experts think of the computers that count your vote.

2. About our study

2.1. The equipment we analyzed

This analysis deals with the Nedap/Groenendaal ES3B voting computer. Together with the ISS election management software, this computer is sold by the cooperating companies Nedap (who mainly do the hardware) and Groenendaal (who mainly do the software).

We started analyzing, on August 23rd of 2006, an ES3B voting computer, a reader unit, the mechanical keys needed to operate the system, two ballot memory modules and the accompanying ISS MS-Windows software. We got all this equipment on loan from municipality A in The Netherlands. Because we were supposed to return the equipment by the end of September, we kept looking for more computers and accessories. On september 6th of 2006, we found municipality B willing to sell us two ES3B computers, another reader unit and two more ballot memory modules. All machines have current software (ES3B 2.12 on two of the voting computers, ES3B 2.11 on both reader units and one voting computer). To the best of our knowledge, these are the exact machines and software versions that 90% of The Netherlands votes on, and at least some of the computers we now lawfully own were in use at elections up until the 2006 local elections in The Netherlands. Barring intervention after publication of this report, we fully expect the machine we gave back to municipality A to be used again in the upcoming elections.

2.2. Early publication because of time constraints, work continues

Because we wanted our results to be available before the November 22nd of 2006 Dutch national parliamentary elections, we were in a hurry. This preliminary report is the result of a month's worth of work. Much more research needs to be done and there are very definite open questions. One of the larger gaps in this report for instance is that we took only a very cursory look at the ISS (Integraal Stem Systeem) Windows software that is used to administer the ES3B voting computers. We have chosen the attacks we implemented based on the limited time and resources available to us. Throughout this paper we will detail a few other attacks that, although we did not implement them, are practically feasible to the best of our knowledge.

As we are writing this paper, further research on the ES3B continues. Our website at www.wijvertrouwenstemcomputersniet.nl/Nedap provides more detailed technical information that would be impractical to include in this paper, such as system memory maps, quite a few programs and many more photos. If you wish to do serious follow-up research involving the ES3B, feel free to contact us.

2.3. Quick fixes?

As a result of earlier press-attention given to our campaign (but before these specific vulnerabilities surfaced), the Dutch government announced some measures to "help increase public trust in an already sufficiently secure electronic voting system". According to the interior ministry, the devices will be sealed, the software will be "extra protected" and certification company TNO will perform extra checks on the software, all before the upcoming election. Details on how these measures will be implemented are not available at this time.

Given the proximity of the upcoming elections and the obvious severity of some of the vulnerabilities discussed herein, we decided to include in this paper as much as possible of our thinking regarding these and other countermeasures.

3. Mechanical keys

3.1. Trust placed in mechanical locks

Dutch election law requires physical keys to be used as part of an electronic voting system. The entire legal framework surrounding voting computers sees these physical mechanical keys as an integral part of the security of the voting process. Dutch election law and regulations makes frequent mention of these keys[1]. The law regulates that voting machines are built such that voting cannot happen without the presence of a key and the law even stipulates that the chairperson of a polling station puts the key in an official sealed envelope after the election is closed. Nedap/Groenendaal, in their 2006 election newsletter, state "programming unit and key as well as ballot modules need to be stored in the safe" [2].

As we can see, both regulators and implementers of electronic voting place trust in traditional mechanical locks and keys and want users to adhere to strict procedures regarding key management.

3.2. Chosen system

The key system chosen by Nedap for both the locks on the voting computer is the "C&K YL Series 4 Tumbler Camlock". This lock always comes with the same key (marked "A126"), which probably explains why the same key is used on all 8000 ES3B machines throughout The Netherlands. Spare keys can be ordered separately online for roughly a Euro each by searching for the product number: 115140126. We ordered, payed for and were subsequently supplied with 100 of these keys without any problem. According to the product datasheet[3], typical applications for this lock include "copy machines and office furniture". Even if spare keys were not so readily available: this is quite literally the type of lock we can open with a bent paperclip.

The reader unit has, as stipulated by law, a lock with a different key for the slot marked 'programming' (it is marked "A154"), which is used to erase the ballot memory modules and to write new candidate lists to the modules. The key is of the same insecure type and the we expect it to also be the same all over the country.

3.3. Conclusion

Even when taking into consideration that the law does not say the physical locks needs to be of decent quality, we feel this lock is obviously grossly inadequate given the trust placed in it. Either this "toy lock" needs to be replaced by a real lock, or the law needs to be rewritten such that it doesn't inspire confidence where none is warranted.

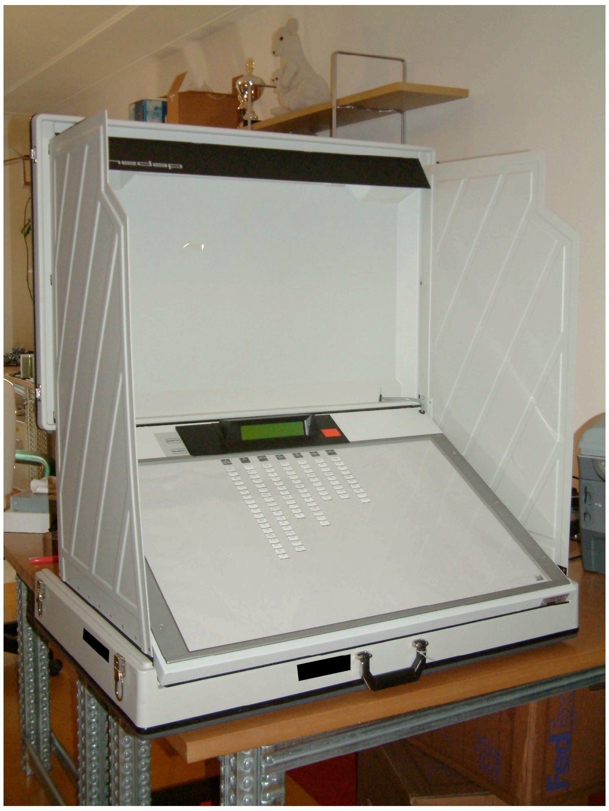
[1] Voorwaarden voor stemmachines 1997: art 3 lid 2, art 5 lid 2, art 6 lid 4, art 7 lid 2.
Kiesbesluit: art. J27 lid2, art J29 lid 2, art. K1.

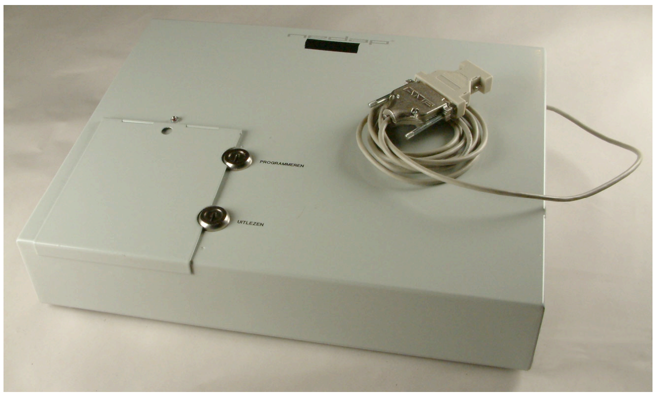
[2] http://www.election.nl/bizx_html/ISS/documents/Verkiezingsbulletin%202%20TK%202006.pdf (Oct 2, 2006)

[3] http://www.ittcannon.com/media/pdf/catalogs/Leaf/YL_1apr.pdf (Oct 2, 2006)

4. Understanding the ES3B

4.1. The components

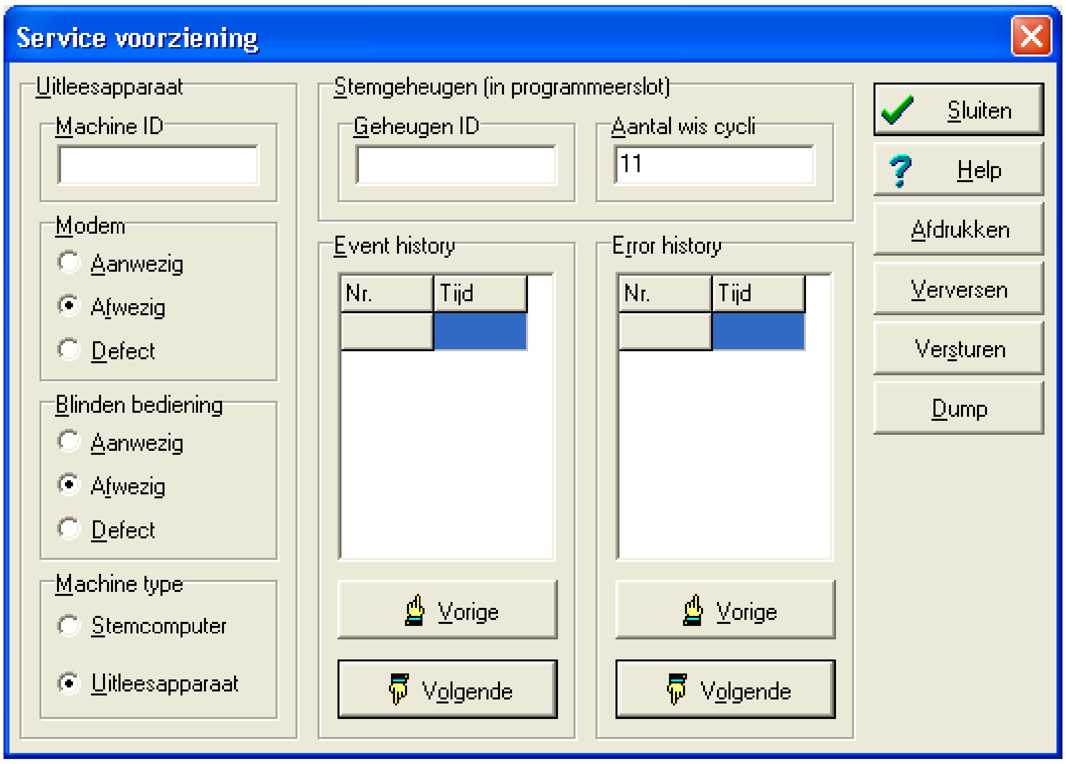




The Nedap ES3B system as it is in use by a typical Dutch municipality consists of multiple ES3B voting computers, at least as many ballot memory modules, a reader unit to be attached to a PC via the serial port and an installed copy of the ISS (Integraal Stem Systeem) software running on a PC under Microsoft Windows. The municipal election officials write the candidate lists into the "ballot memory modules" using the reader unit. Then these modules are installed into the voting computers before they are deployed. After the election is closed, the results are printed by each of the voting computers as a paper backup. The ballot memory modules, which now also contain all the votes, are carried to a central location to be read using the reader unit so the results can be tabulated..

4.2. First contact

We already knew quite a lot about the Dutch Nedap computer from the Irish "First Report of the Commission on Electronic Voting" (2004). As we took it apart, we confirmed that this was indeed a system built around a 68000 processor that came with 256kBytes of EPROM, 8 kBytes of EEPROM, 16 kBytes of RAM, two 6850-based serial ports, a printer port, two screens (4 lines of 40 characters on the voter display, 2 lines of 40 characters on the small election official console) attached to a cable. After taking the system apart, photographing everything and dumping the contents of the EPROMs, we had to put it back together again. First we had to create a working setup to work with the system in normal operation. We installed the ISS software on a notebook PC running Windows XP and we hooked up the reader unit. After some experimentation we could configure a new election, parties, candidates and a polling station. We could then program the ballot memory module that carries the list of parties and candidates to the voting computer. After we inserted it in the voting computer we could cast votes and we could close the election and print the results. After removing the module and placing it in the reader, we could see the ISS software list the votes.



Uitleesapparaat

Machine ID:

Modem: ○ Aanwezig [✓ ● Afwezig] ○ Defect

Blinden bediening: ○ Aanwezig [✓ ● Afwezig] ○ Defect

Machine type: ○ Stemcomputer [✓ ● Uitleesapparaat]

Stemgeheugen (in programmeerslot)

Geheugen ID:

Aantal wis cycli: 11

Event history

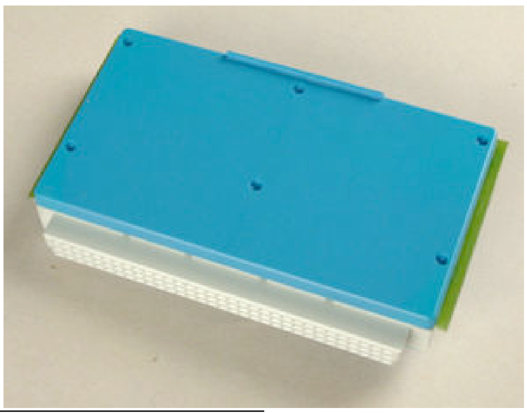
Nr.	Tijd

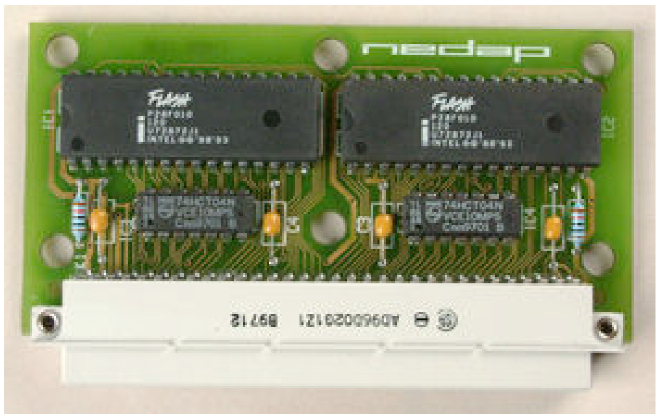
Error history

Nr.	Tijd

4.3. Maintenance mode: "GEHEIM"

The ISS software has a 'maintenance mode' that is supposed to be only accessible to members of the "verkiezingswacht", the Nedap election-day helpdesk. You need a password to get the software in this mode. A quick look in the binary revealed this password to be "GEHEIM", the Dutch word for "SECRET". The maintenance mode, among other things, allows the helpdesk to read the binary contents of a ballot module plugged into the programming slot of a reader unit. By sniffing the serial commands between the ISS software and the reader unit, we figured out how to issue these commands ourselves and subsequently wrote a program in Tcl that we could use to read the entire contents of a ballot memory module.[4]





[4] We used the same code to read the original contents of the system's EEPROM chip, since the 'R' command allows reading of the entire memory space of the system.

Since we were at that stage able to produce memory images in various stages of an election, we could see what changed between them. This produced an overview of where and how the ES3B stored parties and candidates, as well as how the votes were stored.

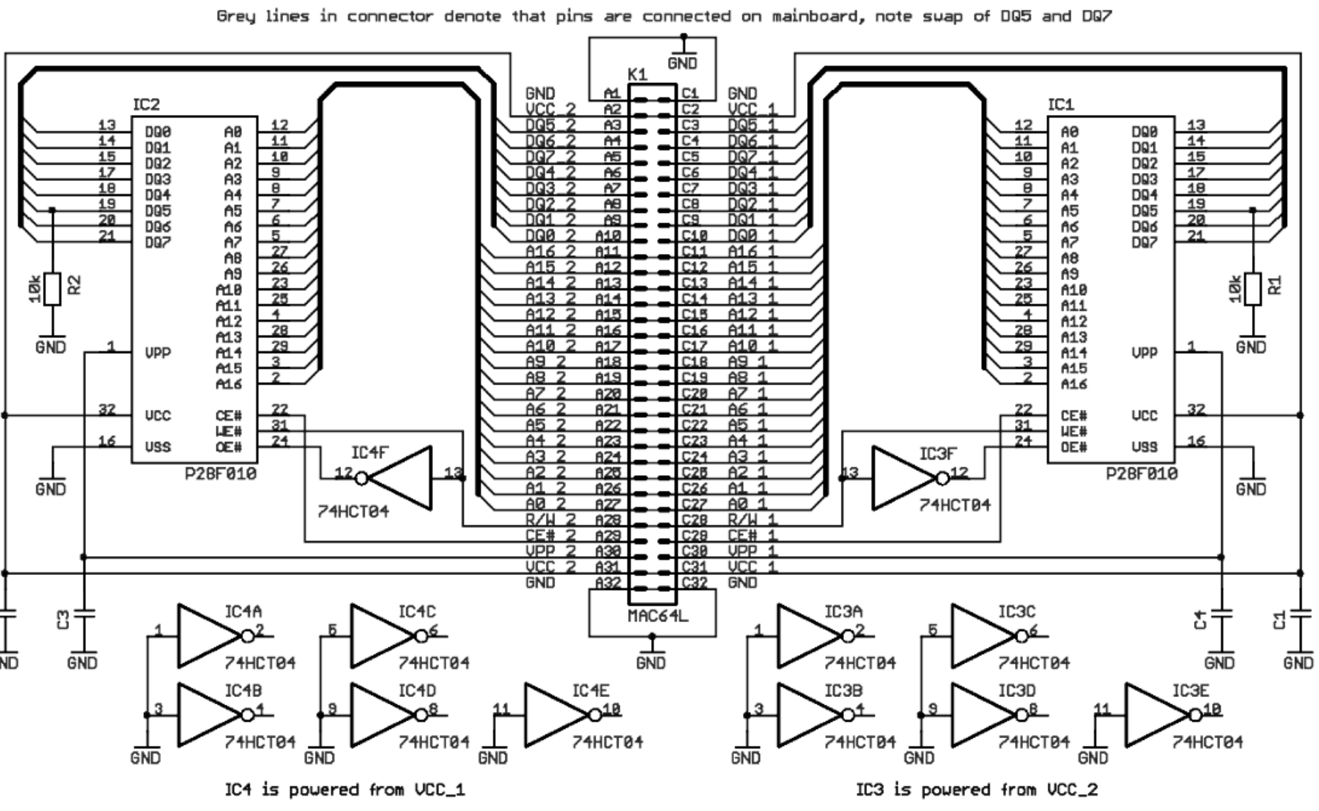
4.4. Trace wires, look up parts, read disassembly, repeat, ...

We used the IDA pro[5] disassembler to look at the compiled binary image contained in the system's EPROMs. Because the hardware is very straightforward and the software very well-written, we were very quickly able to make some sense of the binary. The IDA database contained more and more comments as the hours passed. Most of the IO was figured out by a combination of visual inspection of the printed circuit board and looking at the disassembled code. This was greatly aided by the fact that besides the main components, only 74 series TTL chips were used.

As we progressed, our memory map of the system grew. IO lines to the various keyboards, the key switch, the displays and the printer were documented. We also found more and more about the internals of the motherboard, such as how a watchdog line needed to be pulsed to make sure the board did not keep resetting itself. We also found the small switch on the motherboard that switches the computer into service mode, allowing all of the parameters from the EEPROM (such as the system ID) to be changed through menus on the voter display.

4.5. The ballot memory module

We took apart a memory module and created a schematic. The schematic shows how the two flash chips, which handle the odd and even addresses, both depend on their own hex inverter glue logic chip to make sure that a malfunctioning part will only take out half the memory. All data is written both in the odd and the even half of the module to create redundant data storage.



Grey lines in connector denote that pins are connected on mainboard, note swap of DQ5 and DQ7
IC2
IC1
K1
MAC64L
P28F010
IC4F
IC3F
IC4A
IC4B
IC4C
IC4D
IC4E
IC3A
IC3B
IC3C
IC3D
IC3E
74HCT04
R1
R2
10k
C1
C2
C3
C4
GND
VCC_1
VCC_2
IC4 is powered from VCC_1
IC3 is powered from VCC_2

5 http://www.datarescue.com/

Looking at the schematic, we see that data bit 5 on both odd and even flash chip is pulled down to ground. In addition, bits 5 and 7 have been swapped. It took us a while to figure out that this particular bit needs to be set to one in order to issue the flash erase command on the Intel P28F010 flash chips used, but that it could stay at zero for all other flash programming commands.

We then noticed that this wire is only connected on the ‘programming’ slot of the motherboard, meaning this bit will always be zero in the slot that is used in the voting machine and in the slot that is marked ‘read’ in the reader unit. So only the slot marked ‘programming’ in the reader unit can ever issue the erase command. The swap between bits 5 and 7 places this bit conveniently out of the way. Since the bus on the mainboard is pulled up, the voting machine itself will always write zeroes and read ones on bit 7 of every byte in the ballot memory module.

Candidate lists are stored with the button coordinates of each candidate as well as a checksum, and that part of the memory is subsequently filled with random data. The votes are stored in a different part of the module in a location based on the system timer, skipping to the next one if that location is already taken. Because the pointer in the random locations wraps around after it reaches the end of the module in roughly five minutes, we see no practical attack to get the sequence of the votes cast unless the time each vote was cast is known with some level of precision.

Four copies of each vote are stored, and each copy has “hamming code” error correction added, so storage is extremely redundant. Two of these copies are stored with the bits inverted, which makes sure that subsequent flash operations (which can only turn ‘one’ bits into ‘zero’ bits) cannot change any votes.

Besides candidate lists and votes, the module also stores its ID, and the number of times it has been erased[6]. It also stores the name and date of the elections[7] on the module. After the first vote is cast, it also stores the ID of the voting machine it is in.

4.6. The EPROMs

The software for the ES3B is programmed into two EPROM chips, each holding 128 kBytes. Both chips are socketed, which allow for them to be easily exchanged. The system is built such that one EPROM holds the odd and one holds the even memory addresses. Both EPROM chips carry a small sticker marked with “ES3B”, the software version number, the word “ODD” or “EVEN” and a 32-bit checksum written as hexadecimal characters. This checksum is a simple addition of all bytes in the EPROM. During most of our tests and programming, we replaced these EPROMs by two USB EPROM emulators[8] so we could load our experiment code much faster.

4.7. Running our own software: “Nedap Chess”

It started with what we thought was a very obvious statement. We claimed on our website that the Nedap was just another computer, and that as such it could just as easily be programmed to play chess or to lie about the election results. We didn’t think more of it until Jan Groenendaal, placed a document[9] on the Nedap/Groenendaal website to talk about our website "Wij vertrouwen stemcomputers niet". In it, he says: *“[...] And with regard to the claim that our machine can play chess: I’d like to see that demonstrated”.*

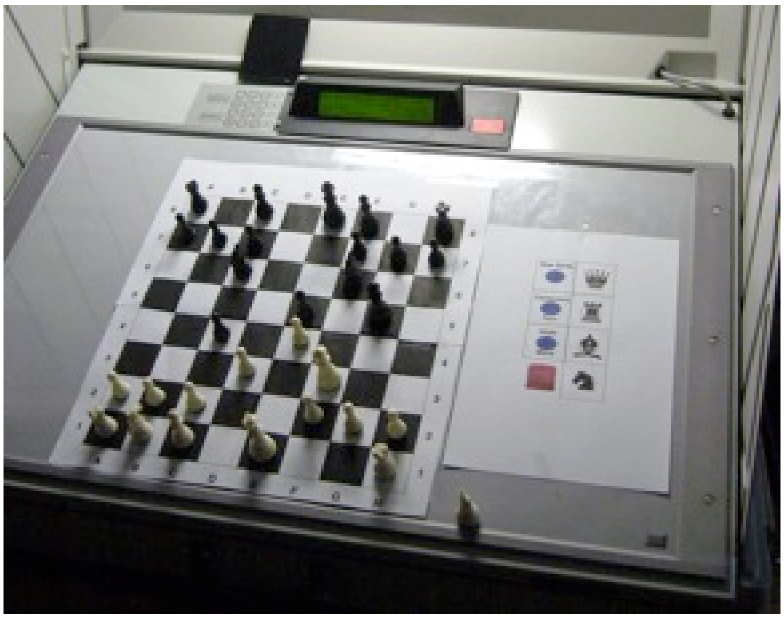
[6] Both can be reset by changing the module ID in the ISS software after going to the service menu.

[7] A maximum of two elections can be done at the same time using the ES3B.

[8] WICE-M4 emulators made by Leap Electronic

[9] http://www.election.nl/bizx_html/ISS/documents/WIJVERTROUWENSTEMCOMPUTERSNIET.pdf (2 Oct 2006)

So obviously, one of our first goals now that we had access to the device was to make it play chess. Apart from proving our point, programming it to do this would also confirm that we knew everything we needed to know about the hardware before getting into the election fraud business. After having learned roughly how the hardware worked we used a gcc 68000 crosscompiler to create a Nedap IO-library containing functions to initialize the system, write data to the display, read the keyboard, and write debug messages to the UART. Together with newlib, a small clib implementation, we then managed to compile and run Tom Kerrigan's Simple Chess Program (TSCP) [10]. This was non-trivial only because we had to squeeze out quite a few tables to make it run using only the available 16 kBytes of RAM. Getting the chess pieces to magnetically attach (the keyboard is mounted at an angle) was also not that easy since the foil switches are stuck to a plastic base. We ended up using using 2 and 5 Eurocent coins underneath the paper, taped such that we could press the underlying foil switches with the edge of the coin.



It knows all the rules and every now and then it can be surprisingly clever for what it is. But in all honesty we have to admit that it does not play chess all that well.

4.8. Conclusion

The Nedap ES3B, regardless of its suitability for elections, is a great system to play with. We much enjoyed our renewed acquaintance with the ESBs vintage electronics. The fact that it uses no PALs or other custom components make it a very easy system to come to grips with. When the ES3B is no longer used for elections, the sudden availability of such a large number of systems might make it an interesting system that could be used to teach young people about the history and basics of computers. We might even put a QWERTY overlay on the keyboard and port a BASIC interpreter to it.

[10] http://home.comcast.net/~tckerrigan

5. Modifying their software

5.1. Introducing "Nedap PowerFraud"

The idea of inserting one's own program EPROMs is not new: again the Irish Commission on Electronic Voting, in their First report (2004: 139) carefully details the attack. It's easier here: the machines tested in Ireland had paper seals that needed to be broken in order to get access to the inner box with the electronics, whereas the Dutch machines at present have no seals. Interestingly, the Dutch interior ministry announced[11] on September 28th of 2006 that all the Nedap machines will be sealed before the November elections. In 2004, in response to parliamentary questions[12] about the Irish report, the responsible minister claimed the Irish situation was fundamentally different so results from the Irish studies did not apply to machines in The Netherlands.

When we started to think about demonstration software that would lie about election results (called "Nedap PowerFraud"), we kept in mind that the system should not lie after an election that was obviously a test of the system. We decided we needed to store the votes and only decide whether or not to perform the fraud at the moment the election was closed, so our program would have as much information as possible to make that decision. Since the voting computer itself cannot issue the flash erase command needed to erase the ballot memory module, and since the votes are stored both as-is and with all bits inverted, we saw no way to change votes already stored in flash. Hence we needed a mechanism that would store votes, so that it could write these to the ballot memory module later, either for the candidate for which they were intended or for the recipient of the fraudulent votes. Since the voting machine could suffer from a loss of power, we would optimally like store these stolen votes in non-volatile storage. Since the flash on the ballot module was ruled out because we could not erase it, we opted for the on-board EEPROM.

The ES3B's EEPROM is normally used to store a few system configuration parameters (such as the device ID) and some settings (such as whether or not the keyboard beep is on). But most of the space is used for two circular buffers holding the event log and the error log of the device. In these logs, the device keeps the system time and a number for each event or error that occurred. Since system time always starts at zero, these times are not as helpful as one might think, they only represent an offset relative to the last processor reset. We updated the circular buffer routines that deal with the error log to shorten the number of entries, making space for our stolen votes. Since our new table didn't have space for all possible candidates, we steal votes only from the number one on each list. Since the majority of voters pick the first candidate on a given party's list, this is quite acceptable. We also store the ID of the current ballot memory module and the date of the election, so we know whether to keep or delete any stored stolen votes when the device wakes up.

We then built "hooks" into the regular ES3B code. Every time a voter casts a ballot, our code generates a random number between 0 and 100. If the number is below the programmed percentage of votes we want to steal, that vote is not written to the ballot module but one is added to the corresponding 16-bit number in EEPROM. At the end of the election, our software determines whether this was a real election or not. It then proceeds to, either honestly or fraudulently, quickly write these votes into random locations in the ballot module, just like the real software does.

To determine the recipient of the stolen votes, PowerFraud does a case-insensitive match of all party names with a programmed string. If it finds a match, that party becomes the recipient of the stolen votes. This allows for the fraudulent EPROMs to be inserted long before the candidate

[11] http://www.minbzk.nl/grondwet_en/kiesrecht/nieuwsberichten/stemmachines_nog

[12] Haverkamp en Spies, CDA, 1 april 2004, KVR20183 / 2030411850 / 0304tkkvr1453 / ISSN 0921 - 7398

lists are known, and it allows a fraudulent ROM to perform the same fraud year after year, even though the relative position of the party on the keyboard changes. It is significant to note that the Dutch interior ministry assumes this to be impossible. A recent statement[13] says: "Fraud during the production of voting machines does not make sense because the lists of candidates are not known then."

5.2. How do you find out the software is lying?

5.2.1. Parallel testing

Parallel testing of voting computers is based on the notion that although the voting computer is a black box, one can test its functionality by presenting a situation that is indistinguishable from a polling station at the inputs of the computer while keeping a careful count of all the votes that go in. To increase confidence in DRE-class voting computers, officials would randomly select voting computers to be removed from the pool on the morning of election day, and replaced with other machines. These randomly selected machines would then be used for such a test election. At the end of the day, the totals should match with the votes that were input.

There has been some work done in the US regarding the number of machines that would need to be tested to achieve a certain level of certainty that the election was honest. First a desired level of confidence would need to be defined, and then the calculations would need to be done in order to calculate the number of machines that would need to be tested for that level of confidence. Subsequently, one would need to define what would happen if discrepancies were detected. The "Oh my God, it is not being honest" at the very end of a busy election day would need to be translated into something more legally binding that allows election officials to call for a new election or keep the results from becoming accepted until more is known. The exact procedure may need to be codified as part of the election regulations.

Assuming these hurdles are taken and a system of parallel testing is in place, the author of the vote-stealing software can perform quite a few tests that would discriminate between a real election and anything but the most rigorous and disciplined parallel tests. The Brennan Center for Justice at New York University very recently published "The machinery of democracy: protecting elections in an electronic world" (2006), which is by far the most detailed work to date to deal with procedures for parallel testing of election systems. It isn't all that optimistic about the method:

> *"However, even under the best of circumstances, Parallel Testing is an imperfect security measure. The testing creates an "arms race" between the testers and the attacker, but the race is one in which the testers can never be certain that they have prevailed." (Brennan Center, 2006: 88)*

The current revision of PowerFraud has a user-selectable minimum number of votes that need to be cast for the election to be seen as real. And although not a true parallel testing counter-measure, it also allows the setting of a minimum number of votes a party must have before stealing from it. This makes sure PowerFraud never inadvertently steals a party's only vote in a given polling station. If it did, it would alert at least one voter that something was amiss.

Next versions of PowerFraud will allow the setting of the minimum time an election should last and we might also incorporate some statistics on the random distribution of the time between votes and the time between the pressing of the vote-release button, the candidate button and the "cast ballot" button. At that point we think parallel testing becomes a job for a robot that presses the

[13] http://www.minbzk.nl/grondwet_en/kiesrecht/nieuwsberichten/stemmachines_nog

buttons because the key intervals of utterly bored human testers are very likely to exhibit statistically improbable timing similarities.

A future version of PowerFraud will also offer a “magic button” function. What this does is allow any voter during the day to press a previously-configured key combination on the voter keypad, followed by the keys needed to cast a vote. The device will then store the party that received that particular vote as the recipient for all the stolen votes, and it will not perform any vote stealing unless the magic button was pressed. This would be impossible to catch using parallel testing. And although post-election examination of the EPROMs would show that a fraud *could* have occurred, it would be hard[14] to detect whether it actually did and which party received the stolen votes. This scenario, although impractical in national elections, would be very practical in a town of, say, ten to fifteen thousand inhabitants where the attackers could send one voter to each of around ten polling stations to press the magic button before they cast their vote.

5.2.2. Verifying the contents of the EPROMs directly

If one were to take the EPROMs from an ES3B and put them in an EPROM reader coupled to a computer, one could compare the contents to a known to be good image of the software to verify that the software has not been tampered with. One could also compare generated checksums, as long as they are not the checksums that Nedap uses. They are simply a 32-bit hex representation of an addition of all the bytes in the EPROM. Only cryptographic hash algorithms such as SHA-256 generate a checksum that is secure against an attacker trying to create a fraudulent EPROM with the same checksum.

The ES3B also allows its memory contents to be read through the serial port. However if this were ever used to test the authenticity of the EPROM, it would be easy to make the fraudulent software lie about the contents of certain memory location through the serial port. Besides: the serial port is only available on the mainboard. One might as well test the actual EPROM contents if the inner box needs to be opened anyway. We can envision a small hand-portable device that would clip over the EPROMs while they are still in their sockets, quickly testing their contents using a small built-in micro-controller. Of course the next problem would be whether or not that device and the people handling it are trustworthy.

5.3. Making sure attackers cannot install software

Given that we have no easy way to tell whether manipulation software was installed, we will need to explore ways to make sure it does not get installed in the first place. Please note that the chapter “On security and elections” explains why fixes along these lines may actually decrease security depending on your viewpoint.

5.3.1. Fix the ES3B so it does not run untrusted software

We do not believe the “anyone can insert their own EPROMs vulnerability” can be fixed in the current design. In order to fix this, one would ideally want a system in which a program already in the processor domain can use the presence of cryptographic signatures to evaluate whether or not it wants to execute/install a newer program. The Nedap ES3B is in essence a very expensive 1980’s home computer. It is not a system that can be retrofitted with the same security-enabling features that today’s low-cost mobile phones, pay-TV decoders and game consoles come with.

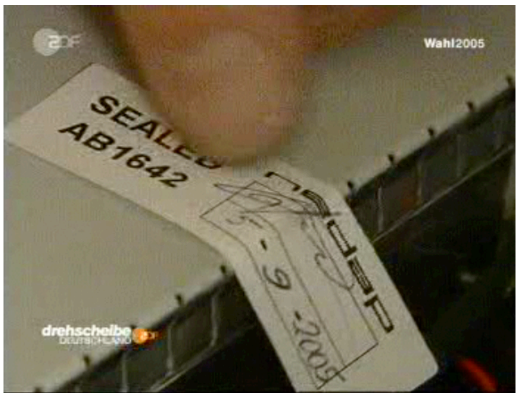
[14] Since the compromising data would have been erased from the EEPROM, it would involve reading past contents of the EEPROM. Not impossible, but we expect this to be costly and time-consuming.

5.3.2. Upgrading physical security

One line of remedies would make sure an attacker could not get surreptitious access to the devices. This can be done by upgrading the physical security of the storage locations and transport logistics. Since this attack against the outcome of the election hinges on physical access to the devices, it would seem prudent to deny access to unauthorized individuals. Given that at least some of these computers are currently stored under rather abysmal security conditions[15], this would add a great deal of security against outside attackers. However it would not significantly increase security against attacks performed by insiders. In situations where human guards are employed, the added security would even increase the number of insiders that have convenient round the clock access to the computers.

5.3.3. Tamper-evident seals

One could also add tamper-evident seals to the inner box holding the electronics. This only makes sense if one tests the contents of the EPROMs first to make sure the attackers did not get there first. This would make it harder for attackers to get to the EPROMs after the seals are applied without this being detected. Also note that applying seals assumes the person or persons applying them would have to be ultimately trusted because they are in a perfect position to swap the EPROMs first. Since a wide range of people would need to properly authenticate and inspect these seals under non-ideal conditions, the value of these seals in adding more than token security can easily be over-estimated.



Note that the German version of this device, the ESD-1, apparently has a seal on the inside, covering the internal electronics housing. Even if one accepts the limitations of sealing listed above, this seal is grossly insufficient because it is truly trivial to bypass. It appears printed on paper and as far as we can tell uses none of the existing tamper-evident technologies. Given that the Dutch interior ministry has announced that all existing machines will be sealed before the next elections, we believe it’s significant to notice a pattern here: tell Nedap they need to incorporate a lock and they pick the cheapest lock imaginable, tell them to put a seal on it and they laser print their company logo on some paper labels.

5.4. Conclusion

The parallel testing method at best presents the testers with an arms race and the other methods appear to exclusively address the threat posed by outsiders. We claim that it is highly questionable whether a sufficient level of over-all security can be accomplished by employing any combination of counter-strategies currently being discussed. We cannot think of other measures that would mitigate this problem.

[15] As documented in the “EénVandaag” October 4th 2006 news television broadcast. It documents how the 400 or so Nedap ES3B voting computers serving the city of Rotterdam are stored in an old building on a somewhat shady industrial lot with no alarm or other security measures.

6. Compromising emanations

6.1. Spurious emissions

Many electronic devices transmit radio signals, even when they are not intended to do so. In the case of computers, such transmissions often leak information about what the computer is doing. The military and intelligence communities have known this for many years and are actively exploiting this against adversaries as well as in shielding their own equipment. We decided to take a look at the radio spectrum emanating from our machines to see of we could determine what they were doing by looking at the transmissions. For this purpose we used an AVCOM PSA 65A spectrum analyzer, as well a number of handheld and tabletop receivers and various antennae.

Please note that although it is probably legal in The Netherlands to look at the emissions from one's own Nedap, it is probably illegal to interpret signals from one that is in use at an election.

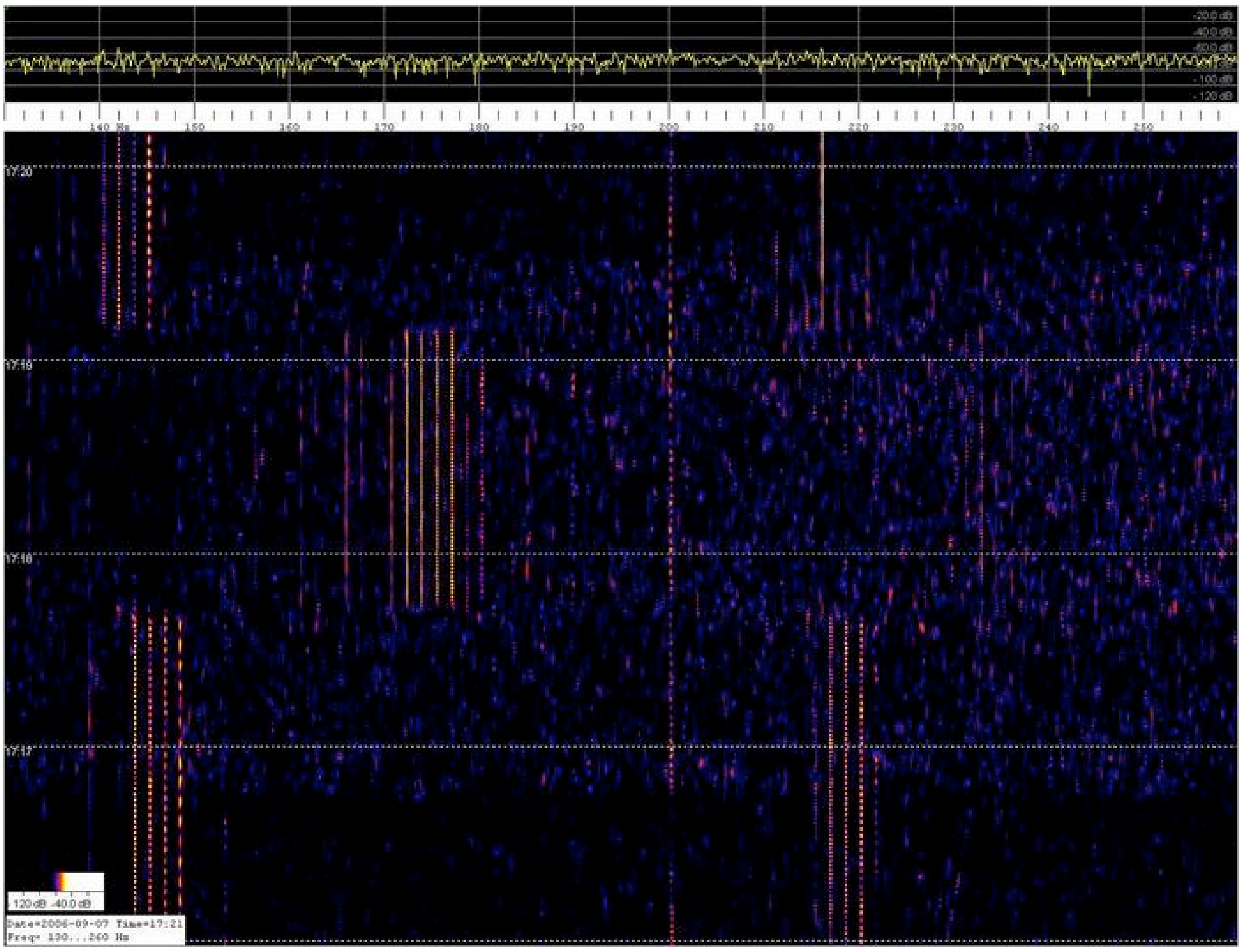
6.2. Special characters and the display refresh frequency

After finding a number of 'empty' signals that do not appear to contain much information, we found one that could be received at many frequencies. Among many other frequencies, it appears at around 36 MHz, 38MHz, 58.3 Mhz and 150 MHz. In both AM and FM, the main energy is at the refresh frequency of the display. The signal is a loud buzz, usually humming at roughly 72 Hz.

The LCD voter display on the Nedap ES3B consists of 4 lines of 40 characters each. Each two lines are driven by a separate Hitachi HD44780 controller. The computer uses a parallel bus to tell the display what to display and where to display it. The controller has its own built-in character set which contains the standard ASCII characters plus, depending on the model, a choice of either Japanese or European extended characters. It can also display up to eight different characters that are not in this character set. To use this feature, the computer must first issue a special command sequence to tell the display controller what the character looks like. For some reason (we suspect cost) Nedap decided to use the Japanese version of the display controller. But they have written software to display any accented or other non-ASCII characters in party and candidate names by first defining special characters for them.

It would appear that if a special character is displayed, the controller has to do extra work every time the display is updated. This causes the display refresh frequency to drop from 72Hz to 58 Hz. The difference between these two frequencies can be determined by ear. In The Netherlands, the name of the major political party CDA is written in full on the display when the voter chooses any CDA candidate: “Christen Democratisch Appèl”. So using only a simple scanner or short-wave receiver, we can tell whether or not a voter is currently voting for a party or candidate with an accent in the name. In Germany for instance this would yield a “Grünen detector”, although the much more frequent use of non-ASCII characters in German names would diminish the selectivity somewhat.

We have observed large signal strength variations between the three devices we have tested this on. In all cases we could receive the signal at a few meters. In one case we could receive the signal up to 25 meters away. Note that a signal like this can be filtered from noise long after the unaided human ear stopped hearing it, so range can be significantly extended using digital signal processing. When experimenting with software to detect these two tones, we noticed that filtering for 216 Hz and 232 Hz respectively, each with a bandwidth of 10 Hz, seems to work better than filtering at the base frequencies of the audible tones. We also noticed energy present at 3845 Hz when the vote-button is pressed.



The image above shows a spectrum waterfall display[16] of the received audio signal. The difference between a candidate from the CDA (middle) and any other party (top and bottom) is clearly visible.

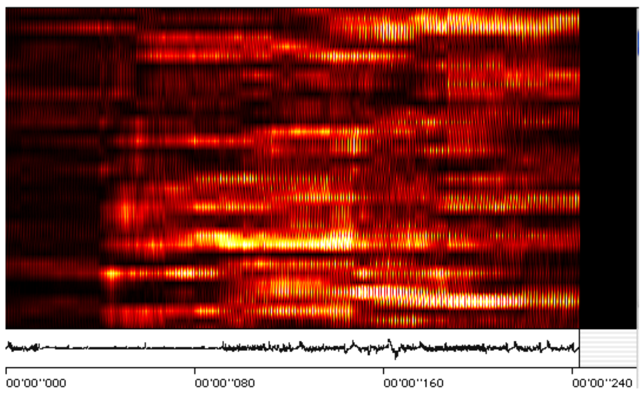
To fix this, the display update frequency always needs to be the same. The display routine in the ES3B's software could easily be modified to always display at least one special character on the display. To make sure this is not visually disturbing this could for instance be a space that is actually defined as an empty special character.

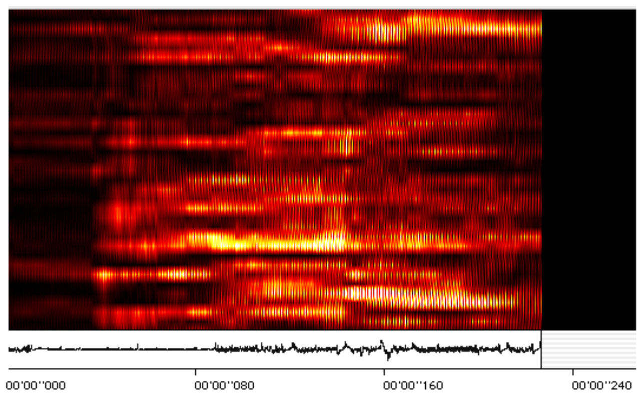
6.3. Fingerprinting display data bursts

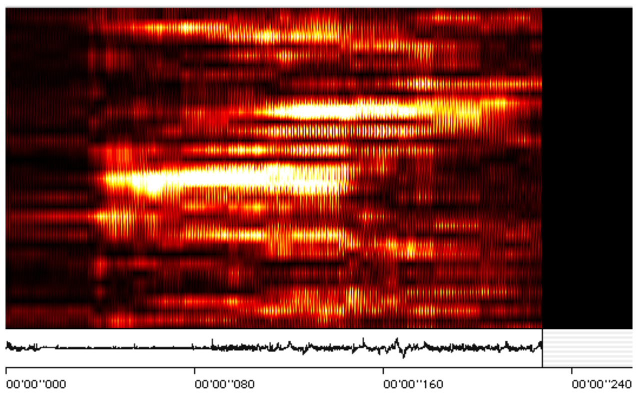
On the same frequencies as the above refresh signal, we can also see a short data burst as the computer writes data to the display. Through the audio path of a scanner, we can make out clear differences for different candidates. Since this signal is generated by a parallel bus, it will likely be hard to decode precisely what is being written the display, especially when one has only a narrow-band audio signal like we did for our experiments. However: we can easily profile all the bursts for the various candidates and simply match the received signal to all known candidates.

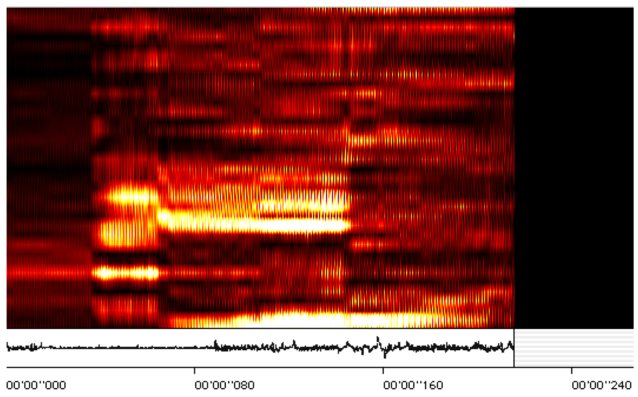
The four spectrum images on the next page show this burst, roughly 200ms in length. The upper two images were received when selecting the same candidate, the lower two were made after selecting different candidates.

[16] Made with SpectrumLab software



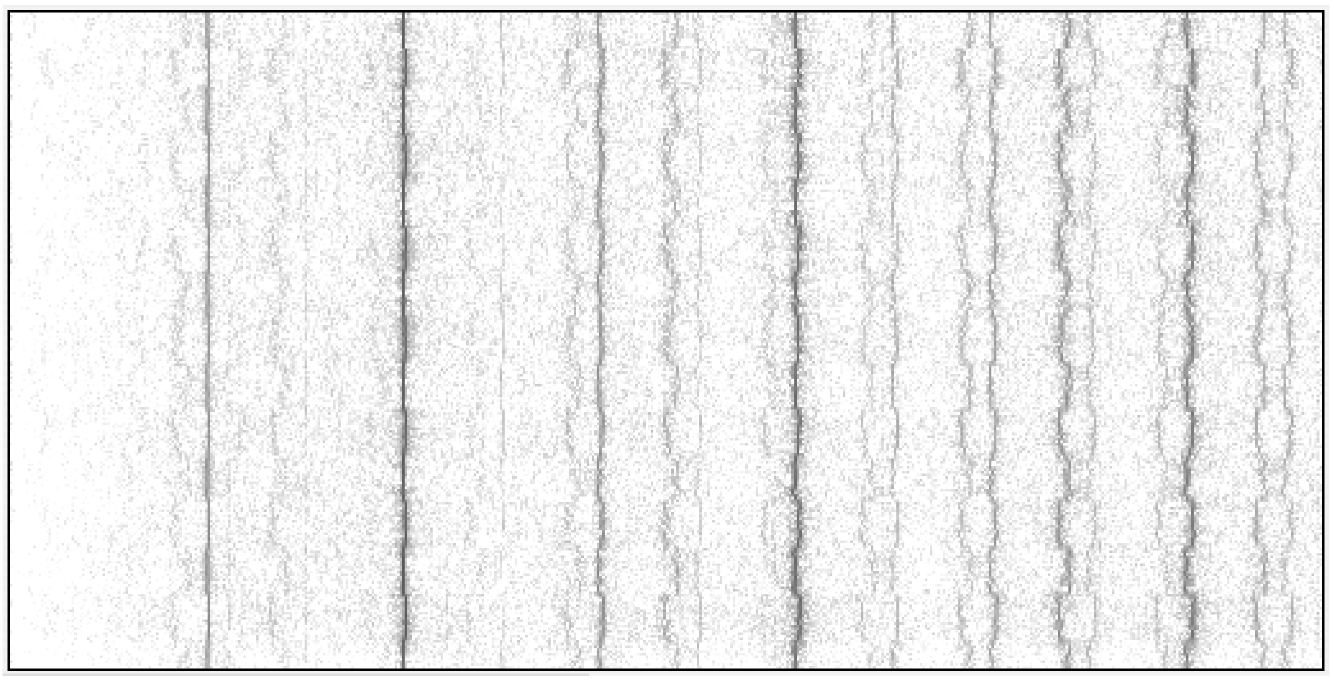






6.4. Listening to the display itself

We did a test while we had our own software display three different screens with a few seconds between them. First we showed the left half of the screen filled with "@" characters, then we showed the right half filled with "." characters and then we showed an empty screen. The spectrum plot shown below shows differences, meaning something can be said about the static contents of the display without listening to the communication between the computer and the display, which happens only briefly. We expect more receiver bandwidth and more processing power would be needed before one can profile display contents in this way.



6.5. Determining impact - defining what constitutes a valid attack

For an attack against ballot secrecy to be practical, we deemed it would have to be significantly easier to perform than obvious attacks such as mounting a small hidden camera that would allow an attacker to see the display of the machine. It is easy to see that much more study is needed: we have only tested the radio emissions on the three different Nedap computers that we had access to, and it may well be that many computers in the field radiate much more or less than any of the machines we looked at.

6.6. Conclusions

The first attack, the "CDA detector", is practical by any definition since it can be performed by ear and takes only a small and cheap scanner or short-wave receiver. If the ES3B were to be used this election, this problem would need to be fixed immediately. Alternatively, if developing and distributing the simple fix needed is not feasible before the upcoming elections, assuming there are no legal impediments, CDA's name could be changed such as to not include the accent.

Fingerprinting display data bursts takes some initial work to create software, but is still very feasible and cheap to perform once the software exists. Given that this attack would yield the exact candidate and party that is being voted for and that countermeasures are fairly straightforward to implement, we feel it would need to be protected against also.

Listening to the display itself would appear to need both a much larger bandwidth to be received and much more digital signal processing. And even then, the range would likely be more limited than it is for the two previous attacks. Within the limits of this preliminary study, we tend to scale this attack below our threshold for a valid attack, although further research as well as the upcoming availability of cheap and portable software defined radio equipment may change this.

It is remarkable nobody appears to have ever tested for any spurious emissions, or thought to include specifications regarding compromising emanations in the legal requirements. It would seem obvious that new versions of regulations regarding voting computers specify a level of shielding that prevents any such problems.

7. Other practical attacks we can think of

7.1. The screen and keyboard man-in-the-middle attack

The voter display on the device consists of 4 lines of 40 characters, controlled by a Hitachi HD44780 controller that is attached with an 8 bit wide bus and some control lines. The keyboard is a simple passive matrix of foil contact keys that is scanned by the computer. One can envision a small board that would be places inside the (rather spacious) keyboard housing and that would terminate the cables from the motherboard. It would then have new cables to plug into the existing keyboard and screen circuit boards. If this board would have, say, a commercial off-the-shelf FPGA with built-in microprocessor, it could pretend to be a keyboard and a screen towards the computer and pretend to be a computer towards the display and the keyboard. Since our little board learns the names of parties and candidates the first time a voter votes for a given candidate, it would have all the necessary information to press the wrong button towards the computer and still present the voter with a readout that reads to chosen candidate. It could do the same level of parallel testing countermeasures that PowerFraud does except it would have a real time clock to aid in detection of parallel testing.

Since none of this would happen inside the main computer housing, placing a seal on this housing would not help against this attack. The sheer number of small torque screws on the keyboard make this manipulation harder, but they also make inspection to check for this attack harder. We've never really played with FPGAs much, so if there is any doubts raised over the practicality of this attack, we might have a go at it.

7.2. Placing a micro-controller in the ballot module

A ballot module consists on a simple circuit board with a few holes that allow the blue plastic shell to be snapped on. One could quite easily create a circuit board that holds a processor, say an Atmel AVR family micro-controller, quite possibly equipped with some external (modern) flash or battery backed-up RAM. This module would store everything honestly until the vote is closed, which it knows because a byte of 0x00 is written in one particular location on the module. At that point it would decide whether or not this was a real election and act accordingly, after which it would allow its votes to be read by the computer for printout. This module could again have a Real Time Clock, allowing for more sophisticated parallel testing detection than would be possible inside the computer itself.

7.3. The ballot manipulator (not a real attack in The Netherlands)

One could easily create a small circuit board with a battery and a micro-controller with sufficient GPIO pins (such as the Atmel ATmega 128) that has a male DIN41612AC connector on it. A program in the micro-controller could manipulate at will the contents of any ballot module that was plugged into it.

Using this circuit, anyone with access to a module as it is being transported to the reader unit and the computer running ISS can change the votes on the module. We realize this is not a true attack in The Netherlands, because it would not alter the data on the print-out, which has already been made at that point. But in other countries, such as Ireland, where the counting of the votes on the module was supposed to happen at some location other than the polling station this attack may very well be of practical value. The First Report by the Commission on Electronic Voting (2004:140) also details this attack.

8. On security and elections

8.1. Security By Obscurity

Cryptography is the science that deals with secret codes. And in cryptography, Kerckhoffs' principle was stated by Auguste Kerckhoffs in the 19th century: a cryptographic system should be secure even if everything about the system, except the key, is public knowledge. Renowned security expert Bruce Schneier takes Kerckhoffs' principle to have meaning outside of cryptographic systems when he says [17]:

"Kerckhoffs' principle applies beyond codes and ciphers to security systems in general: every secret creates a potential failure point. Secrecy, in other words, is a prime cause of brittleness—and therefore something likely to make a system prone to catastrophic collapse. Conversely, openness provides ductility."

The more complex and unchangeable the secrets are that you need to keep, to more prone your system is to catastrophic failure. Conversely, the cheaper and easier it is to change the secrets in a system, the more robust it becomes.

In contrast Jan Groenendaal, the maker of the ES3B software, in 2006 says: [18]

"However Open Source or publishing the source code provides opportunities for dubious characters and unfortunately election and election fraud are both as old as democracy itself. The fact that only few people have this knowledge can also be interpreted in a positive light.

If something goes wrong one quickly knows where to look, and this mere fact is a deterrent for willful manipulation (inside attack)"

This reasoning is a clear example of a controversial design practice often dubbed Security By Obscurity: the inner workings of his system need to remain secret to protect our elections from "dubious characters". Many of the poor design choices that underlie the ES3B's security problems can be excused against the backdrop of the 1980s, when fewer options were available to system designers and many of the present-day security concerns had not yet surfaced. But given that Dutch democracy now completely depends on his technology, the fact that Groenendaal's 2006 viewpoints on security are so far removed from the general consensus in the computer security community is cause for concern.

8.2. Security of what against whom?

In the security community it is not considered very productive to make statements about the security of any system without at least defining what it is we're securing, and what it is we're securing it against. In the case of voting systems, the general concept of security is often mistakenly taken to mean "the system-wide level of over-all security against any attack, mounted by outsiders, that affects the election results". There are many more possible interpretations of security when it comes to elections. The above definition for instance ignores the risk posed by an attack by knowledgeable insiders, or that of an attack that involves only the secrecy of the ballot.

We would argue that in the case of voting systems, the only meaningful security against insiders is to have a voting mechanism of which all the details are published, and that a substantial

[17] In a long interview with The Atlantic Online: http://www.theatlantic.com/doc/200209/mann/4 (2 Oct 2006)

[18] http://www.election.nl/bizx_html/ISS/documents/WIJVERTROUWENSTEMCOMPUTERSNIET.pdf (2 Oct 2006)

portion of the general population is capable of comprehending in-depth. We pose that any other solution creates a situation in which the population depends in essence on reassuring statements that cannot be verified independently. In a country where e-Voting has replaced traditional paper ballots, the level of confidence with which the population views these statements is then by definition the upper bound for the trust the population can have regarding the outcome of any election, and thus in effect a measure for the legitimacy of government. Continued reliance on DRE-class systems will prove increasingly problematic given (among other things) the fluid state of computer security, the increasingly widespread awareness of issues with voting technology and the all too often warranted general distrust of reassuring statements regarding the security of systems whose inner workings are secret.

8.3. More security, less auditability

By adding extra security measures against the over-emphasized threat posed by outsiders, one can actually *increase* the risk posed by insiders. It is not hard to imagine a voting system that would be much more "secure" by today's standards as they would apply to equipment in other fields. Today's mobile phones for instance often combine a processor, execution memory and tamper-resistant key storage to make sure only the manufacturer (who has the cryptographic signing keys) can update the software. These mechanisms can sometimes still be circumvented, but at least they offer a layer of security that is completely absent in the Nedap ES3B. But by adding "security" in this way, the device could also resist any attempts by independent inspectors to see what code it is actually running. So what is a desired feature from the viewpoint of the manufacturer (who is an insider) trying to protect a mobile phone from unauthorized manipulation may thus be a highly undesirable feature from the viewpoint of the concerned voter, who is by definition an outsider.

9. Over-all conclusions

9.1. ES3B insufficiently secure

The extent and impact of the vulnerabilities presented in this paper in our view precludes using the ES3B in any election. Unless anyone comes up with an effective short-term remedy that we have not considered, we believe the short-term remedies available provide insufficient relief.

Let us for a moment assume that the voting public will continue to have trust in DRE-class systems, and let us ignore the problematic lack of transparency and the accompanying threats posed by insiders. Even in such a scenario the mere lack of security features offered by modern processors (such as tamper-resistant key storage and a source of cryptographically strong random) mean the existing Nedap ES3B machines, no matter what software runs on them, cannot ever be made to meet any responsible security criteria for such a system.

9.2. Dutch e-Voting requirements insufficient

It is important to understand that the ES3B meets all Dutch regulatory requirements. These requirements, although very detailed on topics that deal with availability, say absolutely nothing about security against any kind of attack. Without exaggeration, even a brain-numbingly insecure system[19] would would meet Dutch legal requirements, provided the buttons are of a specific size, the computer can withstand moist conditions without presenting a shock hazard and no votes are lost when power fails. Across our research, we noted remarkable similarities to the present situation in the United States. Feldman et al. (2006: 19) write:

> *Despite their very serious security flaws, the Diebold DREs were certified according to federal and state standards. This demonstrates that the certification processes are deficient. The Federal Election Commission's 2002 Voting System Standards say relatively little about security, seeming to focus instead on the machine's reliability if used non-maliciously.*

When a building that was built to meet all regulatory building requirements comes tumbling down without cause, it probably means something was wrong with the requirements. We pose that the same is true here: the fact that the ES3B is proven to be insufficiently secure to the degree shown in this paper while still meeting all applicable legal requirements implies that the legal requirements are grossly insufficient. If we decide we want to continue voting on computers, the legal requirements must at a minimum address basic computer security and stipulate that election results can be independently verified. Existing voting computers will need to be modified to the new standards or replaced if they cannot meet these new requirements.

9.3. DRE voting not given enough thought

DRE systems make us dependent on a single vote count done in software. People cannot see electrons, so they cannot observe a vote count when a DRE voting systems is used. Insufficient thought has been given to the large number of implicit trust relationships that come with DRE systems, such as the near infinite trust placed in the entities building and certifying the systems. In addition, one has to consider the possibility that outside attackers have surreptitious access to one or more devices. Insider attacks are generally thought to be more prevalent, and they are harder to

[19] For the sake of the argument let us assume an unpatched Windows 95 machine with an always-on unencrypted wireless Internet connection, no virus scanner and an early Internet Explorer web browser to cast votes on an ASP script running on an internal unpatched IIS 1.0 webserver.

prevent and/or detect. The latter is especially true in systems of which only insiders know how they work.

We can find no evidence that any of the various trust relationships in the current DRE voting systems in use in The Netherlands have been sufficiently explored. In fact, we cannot find any documents stating that they were considered at all. We predict that very few, if any, cases would warrant the choice for a DRE system after such an analysis was done. Even if a DRE system was built according to today's security requirements, it would still require almost infinite trust in a very limited number of individuals and organizations, which we feel is fundamentally at odds with the notion of a publicly verifiable election.

Given the fact that The Netherlands have a very simple electoral system compared to many other countries[20], the fact that The Netherlands is almost exclusively e-Voting can be excused only by the fact that The Netherlands introduced the computers very early on, at a time when these considerations had not yet surfaced.

Acknowledgments

We thank Job de Haas, Alex Le Heux, Hanno Liem, Peter Knoppers, Tim Kuijsten, Anne-Marie Oostveen, Marcel van der Peijl, Carla van Rijsbergen, Huub Roem, Sander 't Sas, Ferry Stoop, Christiaan Tan, Maurice Wessling, Philipp Wuensche, the Chaos Computer Club Berlin and last but certainly not least the brave heroes at the municipalities that leant and sold us the Nedaps.

References

Brennan Center Task Force on Voting System Security (2006) *The machinery of democracy: Protecting elections in an electronic world.*
Available at http://www.brennancenter.org/programs/downloads/Full20Report.pdf

Commission on Electronic Voting (2004) *FIRST REPORT of the Commission on Electronic Voting on the Secrecy, Accuracy and Testing of the Chosen Electronic Voting System*,
Available at http://www.cev.ie/htm/report/download_first.htm

Ariel J. Feldman, J. Alex Halderman, Edward W. Felten (2006) *Security Analysis of the Diebold AccuVote-TS Voting Machine*. Available at http://itpolicy.princeton.edu/voting/ts-paper.pdf

[20] The Dutch cast only a single one-candidate vote per election, although sometimes two or three elections happen at the same time.