

The Design of a Trustworthy Voting System

Nathanael Paul and Andrew S. Tanenbaum
Vrije Universiteit, Amsterdam, The Netherlands
nate@few.vu.nl, ast@cs.vu.nl

Abstract – After the voting debacle in the Florida Presidential election of 2000 with its now-fabled hanging chads and pregnant chads, many voting jurisdictions turned to electronic voting machines. This transition has had at least as many problems as punch-card systems and added the additional one of making recounts impossible. As a result, many jurisdictions have gone back to paper ballots in despair. We believe that electronic voting can have many benefits including accessibility and usability but requires regarding voting as a system of which the voting machine is only a (small) part. In this paper we describe all the components of an electronic voting system that is practical and difficult to tamper with. We emphasize the importance of systems aspects, defense in depth, and being paranoid.

1. Introduction

Seven years ago, the Help America Vote Act (HAVA) was passed to prevent a repetition of the 2000 Florida Presidential election. With HAVA's funding, states replaced their punch card voting systems and lever voting machines with new electronic voting machines. These new machines were adopted to enhance election integrity by producing an accurate tally while supposedly protecting the votes from being maliciously changed, but the machines are unfortunately still plagued with a multitude of problems [12]. In many cases voting machine errors are not auditable, especially when there is no voter-verifiable paper audit trail (VVPAT). In addition to the typical irregularities and unexplainable errors [11, 43], many of these machines have been shown to be rife with security problems [2, 5, 13, 16, 24, 25, 36]. This stream of problems is eroding voters' faith in voting systems, and election integrity is in jeopardy.

In the past, voting systems were used for a single purpose: determining who got the most votes. As new voting systems have been introduced, designers have added new and previously nonexistent features such as allowing voters to verify their own votes and also the final tally. However, straightforward ways that allow a voter to verify how he voted also allows him to sell his vote and prove it to a buyer, so recent electronic voting research allows verification but prevents vote selling (and its cousin, voter coercion, which is the same thing except the voter is an unwilling participant and does not get paid) [3, 6]. While verification and resistance to vote selling are desirable, their inclusion in proposals has led to complex designs that few legislators or voters can understand. While many of the solutions implementing these features are elegant, the features themselves have little to do with election integrity (e.g., a recently deceased registered voter's vote can

still be cast). In our view, maintaining the integrity of the election is paramount and features achieving other properties are secondary.

Our goal is to design an electronic voting *system* that restores voter confidence through its simplicity and security. The overall design is different than most voting systems as we focus on designing an entire electronic voting system from beginning to end. Although other paper-based systems have recently been introduced [7, 8, 14, 35], these schemes are outside the scope of this paper because of their paper-based design. Our motivation is that a well-designed electronic voting system has several benefits including improved accessibility such as audio for the blind, cheaper and faster reporting of the tentative vote tally, and more flexibility in displaying custom ballots (e.g., ballots in multiple languages, a larger font for the elderly, prevention of overvotes, and feedback on undervotes).

In addition to these benefits, complexity remains as a challenge to a voting system's acceptance. Voters and legislators who do not understand a complex voting system will not accept it. Not only is a simpler system more likely to be understood and accepted, but it should be more robust. Unlike the people running other complex systems (like airplanes), the people running a voting system may want it to fail (i.e., be able to secretly modify the results) for partisan reasons. Because of the challenge of building a dependable electronic voting machine that is resistant to failure (from attack or error), the voting machines must not be able to undetectably alter election integrity. This can be achieved by having the machines print voter-verifiable paper ballots and paper receipts to ensure election integrity is independent of the voting machine's operation [37].

Contributions. We present a transparent voting system from the very beginning of an election to the final tally, specifying exactly how a Trusted Platform Module (TPM) is used, presenting a scheme that enhances registration integrity, and introducing a design that prioritizes election integrity (An earlier introductory version of this work is also available [32]). We have developed a nine-step voting system that takes place from an election's inception to its final conclusion (Section 3). Where possible, we have used standard cryptographic primitives and a TPM throughout the design. While others mention using trusted hardware [10, 21, 39], we specify the TPM's use in detail and take advantage of its existing PKI infrastructure (Section 3). Because of new concern over registration integrity [31, 33, 42], we have also added a new component that better ties registration into the act of voting (Section 3, Step 2). Our verification process is different from most current voting systems as a voter

can easily check if, and how, his or her own vote was counted (See Section 3, Step 9).

2. Assumptions

We make four assumptions: (1) each voting machine has a TPM and a mechanism to perform run-time attestation (current voting machines do not support this); (2) advance voter registration is required (making it unsuitable for some states); (3) voters can use and write down a password established at registration time to be used on election day; and (4) human-readable receipts that plainly show the voters' choices can be printed by the voting machines.

TPMs are attractive for use in voting machines mainly for their hardware protection of cryptographic keys. This work is among the first to explicitly detail how a TPM should be used in an election – specifically in how to handle keys and for software attestation. We do not solve the key management problem, but we offer an approach to manage voting system keys using the TPM's established PKI. In addition to key management, we use the TPM to attest different software used throughout our voting system. By using open source software, and allowing voters to verify that the published open source software is running at the time they vote, people will have more faith that the election is being run honestly [17]. People are more likely to trust a voting system that is more transparent and allows source code inspection.

If a voter, poll worker, or other third party chooses to attest the voting machine software, successful attestation tells him that it is *likely* that the machine is running that software and the machine is not recording private information (We assume that compromising the machine's hardware on which attestation depends is a nontrivial problem). Due to the possibility of hardware compromise, we use paper ballots with paper receipts, and this prevents a machine from undetectably altering a vote as long as the voter checks the receipt. If there is a discrepancy between the electronic and paper record, the paper is the final and trusted result.

We additionally assume that voter registration is required. This rules out using this type of design in a state where advance registration is not required, but states can change legislation to use this system. Recently, the media have given much attention to registration integrity, and many states have had problems in maintaining their registration databases [42]. In 1997, Florida uncovered a corrupt Miami Mayoral election, and they experienced registration problems when they purged their voter database to stop double voting or voting by the dead [29]. Recent problems with the voter database in New Mexico has also cast light on this problem [9]. Our system uses an append-only voter registration database that provides a clear record of all database changes. While using an append-only database is not novel, this is one of the first voting systems to integrate a registration integrity solution into its design.

To enforce registration integrity, part of the registration process requires the voter to create (and optionally write down) a password. Requiring a password is similar to other voting systems that require passcodes for the voters [18]. In our system, a password is required in order to vote (a fail-safe is provided), but this is the only additional burden that most voters will experience. The voter can ignore other slightly more complicated parts of our voting system (e.g., interaction with a TPM via attestation), and this simple password protects votes from being stolen at the precinct.

Recent cryptographic voting research has attempted to solve the problem of vote selling while also providing auditability through receipt verification. Many of the current electronic voting systems do not allow verification (e.g., currently deployed ES&S, Diebold, etc.), and using these systems for an election has proven disastrous [12, 26, 40]. Without the ability of producing a reliable audit, many question the election outcome. Due to these problems in auditability, our system does not equally value vote selling resistance and auditability. The main priorities are election integrity and having voters understand the system.

The lack of auditability has decreased voter confidence in current election systems [4], and many voters are turning to absentee ballots. In just four years, early voting has increased to approximately 30% of all votes, an increase of 10% over the 2004 presidential election [28]. Of these early votes, the available 2008 election data clearly shows that mail-in votes are a significant percentage of all votes cast. With few reported problems of vote selling and so many problems of auditability, we emphasize auditability over vote selling by using human-readable ballots and receipts.

In addition to auditability, human-readable ballots and receipts are essential for building voter trust. If voters and politicians do not understand the system, then they will not have confidence in the system, and recounts will become less meaningful. Our idea of using human-readable receipts is not new [34]. In fact, most believe that introducing the possibility of vote selling makes the voting problem trivial, but it is not. Making a trustworthy electronic voting system (trusted by both politicians and voters) that is both reliable and auditable is challenging. This paper is about the design of such a system.

3. Outline of the Proposed Voting System

Our voting system consists of nine steps, listed below, which take place in sequence during a period that may take up to a year after the election is called or the process started. In this paper, we will use the U.S. names for the officials involved, but analogous ones exist in other democracies.

1. Precinct master key generation and distribution.
2. Voter registration.
3. Proof of registration mailed to the voters.
4. Voting machines are prepared.

5. Key assembly at each precinct.
6. Voters show up and check in.
7. Voters cast their votes.
8. Tabulating the votes.
9. Publishing the result.

Each of these steps has some subtle points and potential for malfeasance or fraud. Some of these steps rely on a Trusted Platform Module (TPM), and we now outline the TPM functionality needed.

Using Trusted Platform Modules for Attestation and Key Management. Our goal is to use an open design in our system to engender trust. We use open-source voting software (currently under implementation), publish it on a website and allow verification of that software. Getting states to use open-source software is a political and legal issue. The technical challenge is to allow voters and others to verify key properties about a machine’s configuration immediately prior to using it for voting, a process called *attestation*.

Our attestation assumptions are:

1. The voting machine *hardware* operates correctly and has not been compromised.
2. The private key of the TPM has not been leaked.
3. All of the software that can potentially execute during the voting process is included in the TPM measurement (described below).

If the assumptions hold true, then attestation shows the machine is running the published open-source software and a successful machine or key compromise is made more difficult (assuming that compromising the TPM is a nontrivial problem). The possibility of successful verification under violated assumptions still exists, and we must also ensure that the machine cannot undetectably affect the election outcome (although violations will hurt other voting system properties including privacy and robustness). To ensure this property, our voting system uses human-verifiable paper ballots and

human-verifiable paper receipts that can easily be checked after casting a vote.

In our scheme, software on the voting machine is verified by computing its hash and then comparing it to the published hash of the open-source code. To perform attestation, we use a new instruction in x86 chips and a hardware device called a Trusted Platform Module (TPM) that is already part of many modern PCs. Although our design uses x86 chips, specifically AMD x86 chips, Intel has similar functionality in newer chips that could be used [19].

In AMD processors that support TPM version 1.2 chips, there is an x86 instruction called *skinit* that cryptographically hashes the contents of 64-KB of memory [1]. This instruction disables paging and interrupts, disables DMA to the 64-KB memory region, verifies that all cores are disabled but the one running *skinit*, runs a hash on the 64-KB of memory, stores the hash in a specific TPM register, and then executes the code stored in the 64-KB of memory. Later, a challenger can ask the operating system for a cryptographically signed copy of the TPM register containing the hash of the 64-KB code. A certificate for the corresponding public key can be provided so anyone can verify the hash of the code. Since only the TPM has the private key, if the signed hash of the 64-KB memory is correct, the 64-KB program, which we will call the *checker*, must have been correct.

We use the checker to verify the entire voting machine software. The checker hashes all of memory (including the operating system), any data that could affect the machine's operation (e.g., ACPI tables and the BIOS system management code [22]), plus the main BIOS, CD-ROM BIOS, and any other BIOSes present. It also keeps interrupts and DMA disabled, so that the attested code never loses control. Once the code is verified, it always remains valid and in control (the machine is not on any network). Once you can be sure that the running software is identical to the published software, the rest is manageable.

In four different parts of our voting system design, we use the TPM to attest that the checker is correct. If the checker is correct and it produces a valid measurement of the rest of the

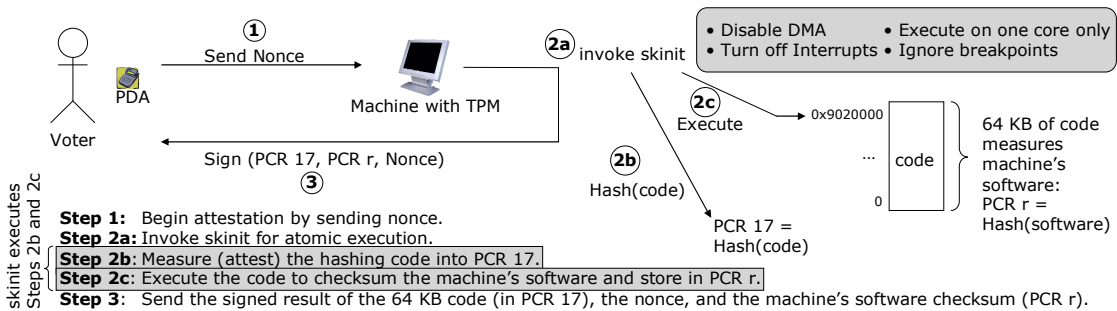


Fig. 1. Code verification using *skinit*. Steps 2b and 2c are executed atomically by the *skinit* instruction.

voting machine software, then we can conclude that the machine is running the published software under our previous assumptions. If attestation fails, then a different device should be used to make sure that the failure is not with the device issuing the attestation challenge.

To begin attestation, the algorithm accepts a random value (a nonce) as input as shown in Fig. 1 (Step 1). It then disables interrupts and DMA to the memory containing the checker just before it executes `skinit` (Step 2a), computes and stores a hash of the checker program in TPM Platform Configuration Register (PCR) 17 (Step 2b), and then executes the checker. Before the checker exits, it writes its result (the hash of all of memory, code, and relevant data) into a different PCR register, `r` (Step 2c). After `skinit` has finished, the machine returns the TPM signature of {PCR `r`, PCR 17, nonce} (Step 3).

From using the TPM to store keys and to help with attestation, the TPM is now a primary target for attack. While the keys reside in a TPM, the manufacturer of the hardware could act maliciously. Even without malicious intent, buggy hardware may yield to compromise. Although we can have independent authorities check the hardware for specification conformance, an examiner may miss a bug or vulnerability. Due to the possibility of bugs, no solution should place all of its trust in the hardware. We provide voter-verifiable receipts and voter-verifiable ballots to protect against both malicious and non-malicious hardware issues and use the electronic count for quick results. Any discrepancies of count are resolved in favor of the paper ballots.

Step 1: Precinct Master Key Generation and Distribution.

Like other voting schemes, multiple keys are needed in the election. Computational load is not an issue (a voting machine can easily handle 600 voters in 15 hours), so public-key cryptography (e.g., RSA) will be used due to its simpler key management.

We use three keys to encrypt and sign voting data.

Keypair 1. Encrypts/Decrypts files on voting and poll worker machines (per precinct)

Keypair 2. Ballot-signing keypair (per voter)

Keypair 3. Software attestation signing keypair (per attestation)

A single key pair (keypair 1) is needed per precinct (typically a school or firehouse with perhaps a dozen identical voting machines) to lock/unlock files on the voting machines and also on the poll workers' machines. The encryption of the relevant election files on all the precinct machines ensures their data confidentiality up to the start of the election. If this private key is compromised, the voter authentication tokens are in jeopardy (the password hash, see below).

For the other two keys, we can use each machine's TPM to generate new keys for each voter. Each ballot is signed by a

unique, freshly TPM-generated key (keypair 2). Another TPM-generated signing key is generated for each voter-initiated request to attest the software (keypair 3). Both signing keys are signed with a freshly generated TPM key, called a TPM *attestation identity key* (AIK), that shows that the TPM is managing the private keys in keypair 2 and keypair 3; we use the machine's single endorsement key (the most trusted key in the TPM) to sign each TPM attestation identity key to show that it is a valid AIK. Each AIK and single endorsement key (EK) never leave the TPM and are not part of the three listed keypairs.

The EK is the foundation of trust in a TPM. For each machine's EK, a certificate is provided to show its authenticity. In addition to the EK certificate, a platform certificate (signed by an independent third party) can be used to show the machine and TPM's conformance to specifications. Using the EK and AIK, the group can then verify the software of the machine before using it to generate keys.

Individuals can inspect a machine's endorsement key (EK) certificate (from the TPM manufacturer, or alternatively, regenerated at this event) to verify a machine has a legitimate TPM. An EK can be issued to the TPM in one of two ways: by generating the EK inside the TPM or injecting the key from outside of the TPM. We advocate the creation on the inside to take away the possibility that someone could get the key before it is injected. Thus, in order to compromise the EK, collusion with the vendor or a compromise of the trusted hardware is needed. This assumes that people trust the certificate authorities and certification processes, and reliable certification processes are in place.

The TPM-created keys do not need distribution (keypair 2 and keypair 3), but the keys that are used to decrypt the data on the voting and poll worker machines (keypair 1) do need distribution, because the decryption key will be distributed later. For California's approximately 25,000 precincts, 25,000 key pairs must be pregenerated, stored, and distributed. We are distributing a keypair per precinct (not per machine) and taking advantage of the TPM's already established PKI for signing. Other alternatives include having one key per county or perhaps one key per state. We felt that having one key per precinct presented the best balance between key management and the impact of a key compromise. At worst, a precinct key's compromise will only affect the voters for that specific precinct.

These 25,000 precinct keys are generated as follows. The Secretary of State chooses a particular brand and model of computer to use (e.g., by competitive bidding) that supports a TPM chip. On a designated day months before the election, he or she invites all the political parties and the media to a public key generation event. Each party may send one party officer and one technical expert chosen by the party.

After attestation by all the technical experts present, the precinct keys (one pair per precinct) can be generated outside of the TPM by the now-verified trusted software. The new

public keys are signed and stored on a notebook computer. Because the machine's integrity has just been checked, the keys are trusted. The private keys are split using any well-vetted secret-sharing scheme. A fault tolerant scheme will likely be needed in the case of someone losing her or her key part, but such schemes are well known.

Each part of each key's secret is written onto some tangible medium such as a contactful smart card (so there are no radio signals to intercept). Smart card reader/writers are available with RS-232C serial line interfaces, which have extremely simple device drivers (unlike USB drivers, which are much more complex) enabling easier code verification. The PC can have a PCI board with a dozen serial lines so many smart cards can be written in parallel. If need be, multiple PCs can be used in a similar way so all the smart cards can be produced in one day, while the political parties' technical experts watch the PCs and each other like hawks.

For the moment, assume each private key is divided into just two parts, *A* and *B*. When all the smart cards for a particular county have been finished, the *A* parts are put into a briefcase and locked and handed to the county's registrar of voters and taken back to his county. The *B* parts are put in another locked briefcase and given to the county sheriff and taken back to the county separately. They are locked in separate safes in different buildings until the election. For extreme paranoia, the keys could be split into, say, four parts each, with the two leading political parties in each county each getting pieces. The key cannot be assembled before election day since the various parts are being held by independent (and potentially hostile) parties. This scheme tacitly assumes that no part of any key is lost during this process, and at least one private key holder does not collude. However, other types of (fault-tolerant) threshold schemes could be used in practice.

Step 2: Voter Registration. Once all the keys have been distributed, voter registration can begin (Fig. 2 shows a voter registration record). If the keys are reused, then the voter does not have to re-register. To register, a voter goes to the county office with the necessary identification as required by state law (e.g., proof of residence). As each voter registers, a record is created for that voter in an append-only file.

To protect against attacks by dishonest poll workers, we add a voter-generated password needed to vote. Since the voter may not trust the county officials with the password, he may bring a device (e.g., a PDA, laptop, or cell phone) with the password preloaded on it. Voters lacking their own device can use the county's computer to enter their password, but then they have to trust the county not to steal it. The voter will use this device to send his hashed password (*not* the plaintext password) to the registrar's computer.

Some voters will pick weak passwords leading to easy offline brute-force attacks. If we use salt values as tra-

Voter ID: 31415926
Precinct: 4072
Name: Mary Hatch
Address: 323 Sycamore, NY, NY
Party: Independent
hash (passwd confounder)
hash ($S_{i1} S_{i2}$)
hash (record database)
Encrypted with precinct key 4072 ()
Encrypted with county public key (X)

Fig. 2. A voter registration record

ditionally done to defend against password guessing, then this does not help with someone that has access to the password database and all the salt values. Instead, we can use the precinct public key to encrypt the password hash with a random value rather than just storing the hash as is normally done. The password hash will not be needed until election day, and the precinct private key will not be ready for decryption until then. Fortunately, this extra security does not add any complexity for the voter. He or she will continue to use his device to enter his password. The difference is that the device then sends $E_k (hash(password) || confounder)$, where *k* is the precinct public key, || is concatenation, and the confounder is a random value that is solely to prevent guessing [15]. The encrypted password remains noninvertible until the secret-shared private precinct key is reassembled on election day.

In addition to these steps, and for defense in depth, the registrar's computer generates a secret for voter *i*, S_i and breaks it into two parts, S_{i1} and S_{i2} where $S_i = S_{i1} || S_{i2}$ (where || means concatenation or XOR). It encrypts S_{i1} and S_{i2} with a county-generated public key and stores $hash(S_{i1} || S_{i2})$. Each of these values are added to the voter's record and will be later used on election day.

Once the new record is ready for insertion, it is immediately cryptographically hashed (with the rest of the entire voter database), the hash is encrypted, and then the hash and the record are inserted into the database. To complete the record's creation, a signed, time-stamped printout of the relevant information is made to record the voter's registration, and the record is transmitted to a centralized state-wide location (complies with HAVA's requirement for a centralized database of all registered voters). Immediately after registering, the voter is encouraged to write down his chosen password for future reference. Later, any voter can check his status by going to a state website as can be commonly done today. This procedure detects dishonest county registrars who discard the registrations of selected voters.

Because a voter's registration information may change (e.g., people may move or die), database modifications will be necessary. To keep the integrity of the already computed hashes, the database records are never modified in place. Instead, when a voter record is modified or deleted, a signed record describing the change is *appended*. In this way, we

can have an audit log of all modifications to the database. Using this registration design, we have distributed the trust among local and state participants, and we have created an audit trail of the registration process. To check added and deleted voters, we suggest using random audits of database records to catch attacks that would register nonexistent or ineligible voters.

This design is open to new attacks. A dishonest registrar could compromise the machine, and record secret information (the secret values, S_{i1} and S_{i2}). To protect against the revealing of S_{i1} and S_{i2} , we could use the voter-supplied device (or the county-supplied device) to contribute to S_{i1} and S_{i2} , but the voter would then have to be able to check if his contributions were used in S_i 's generation. For simplicity, we use the design described, and note that an adversary has little power without the voter password, which even the registrar does not have.

Step 3: Proof of Registration Mailed to the Voters. A few weeks before election day, the county sends to each voter by snail mail a sample ballot and booklet with the candidates' statements, information about ballot initiatives, etc. Many states already do this. However, now, included in the packet is a single-use difficult-to-forge card (e.g., printed on security paper, containing a chip, etc.) that serves as proof of citizenship, residence, and registration, so that those issues need not come up at voting time (because you cannot register without meeting the legal requirements). The card is free, just like the sample ballot, so as not to put a burden on poorer voters. The card will cost the states money, but revenue not spent on registration difficulties can help cover the card expense.

In addition, and most important, the card also contains the S_{i1} generated and recorded at registration time. It could be printed on the card as characters, printed on the card as a bar code, put on a chip etc. (S_{i2} is encrypted and electronically recorded in the database but is not on the card). The card also contains the address of the polling place, the hours it is open, and a reminder to bring your password. The voter will use this card for authentication to a poll worker on election day (Step 6).

The registration mailing has some different attack vectors. Someone could intercept S_{i1} , but this should not be a problem (the voter password that will be required later in the voting process is still unknown). Denial-of-service attacks are still a problem. For example, someone could purposefully (or accidentally) fail to mail out some of the cards, or they could mail out the incorrect S_{i1} . These attacks would be more difficult if it were possible to require multiple people to mail registration cards together (forcing collusion for a successful attack), but having multiple mailing participants may not always be practical. Like current systems, we do not anticipate large-scale problems with the delivery via mail.

Step 4: Voting Machines are Prepared. For each voting machine in precinct i , a file is prepared containing the list of all voters in that precinct (This is why a short plaintext header is needed before each encrypted record). Each machine in the precinct gets the same list so a voter can pick any voting machine and it will have the necessary information. If a voter goes to the wrong precinct, he will have to cast a provisional (paper) ballot since the voting machines there will not have the required record.

Election officials will use the state-wide list of registered voters to build new lists of registered voters for each precinct. This list contains the set of all (encrypted) voter records for that precinct, but a different integrity field will be used in each record for the shorter list (each record's integrity value in the state-wide list is calculated using all the records before it). Because the precinct list is a subset of the entire statewide list, its creation should be done by a group of trustees to protect against precinct list attacks.

After voter registration has ended, the entire precinct list is stored on a read-only medium (e.g., a CD-ROM) that will be used to boot the voting machines in the precinct. The point of encrypting the entire voter file is to prevent anyone from tampering with it while it is in storage prior to the election or in transit to its precinct. A second CD-ROM is also prepared for the poll workers' machines at each precinct. This CD-ROM contains the file containing each voter's ID, name, address, and S_{i2} value (verified by the voting machine to make sure a voter has been properly authenticated). This file, prepared by the registrar, is also encrypted using the precinct's public key to prevent tampering in storage or transit.

Step 5: Key Assembly at Each Precinct. Well ahead of the election, the EK and platform certificate for each voting machine and the public key of the precinct are posted on the county's website. Just before each precinct opens on election day, say at 5:30 A.M. for a 6 A.M. opening, the head poll worker shows up with the county's half of the precinct's private key. He gets it (on a smart card) from the county registrar, who unlocks the safe the day before the election. Similarly, a sheriff's deputy brings the other half at 5:30 A.M. as well. If political parties have fractions of the private key, they also come at this time. Legal sanctions should be in place to encourage showing up on time (to prevent denial of service attacks by shutting down the polling place). This practice is similar to current distribution methods where officials hand-deliver and load ballot information onto the voting machines just before voting begins [20].

Before being booted, the electronic voting machines are inspected for signs of tampering. Alternatively, the machines could be vetted back at headquarters the day before and hermetically sealed in a tamper-evident way. The machines (which have no hard disks) are now booted from the precinct's CD-ROM.

Once the voting software has been loaded, the poll worker uses a PDA to perform attestation, as described above (For additional security we could verify the machine with multiple devices). As usual, the verified software disables interrupts and DMA so unverified software never gains control. Without a network, the attested code will continue execution without interference.

After the poll worker verifies the machine's integrity, the smart cards with the precinct's private key parts are successively inserted to assemble the final precinct key to decrypt the passwords and S_{i2} on the CD-ROM. The precinct key assembly can take place outside of the TPM, because all code on the machine has now been verified. Since the full precinct key was not available in one place until this moment, no one could have meaningfully changed the encrypted values in the voter files during their transport or storage. The last step in getting the precinct ready is verification of the poll workers' machines (done in the same manner as the voting machines). After this step, the precinct is ready to accept voters.

Step 6: Voters Show up and Check in. When the doors open, the first voter approaches a poll worker and hands over the card he was mailed. (In the absence of the card, a paper provisional ballot has to be used.) The poll worker enters the voter's ID in a computer, thus bringing up the voter's (now decrypted) record. The poll worker checks if the name and address on the screen match the card. For additional security, a digital photo of the voter taken at registration time could be included in the computer record and/or printed on the card. (A stolen card is worthless without the password.). The poll worker then asks the voter if he remembers the password entered at registration time. If not, the voter is given a paper provisional ballot. Once such a system is introduced, people will be constantly reminded to choose passwords easy enough for them to remember, like the full name of their favorite cousin.

Then the poll worker uses a bar code reader to enter S_{i1} from the card. The computer then concatenates the S_{i1} value with its stored S_{i2} value to get $S_i = S_{i1} || S_{i2}$. It then creates a voting token (a contactful smart card) containing the voter's ID number, VID_i , and S_i . After the token's generation, the computer re-encrypts the voter's record. The voter is handed the voting token, as shown in Fig. 3, and told to go to any voting machine and follow the on-screen directions.

Step 7: Voters Cast Their Votes. Before starting to vote, the voter may want to verify that the voting machine is indeed running the open source software published on the county

registrar's website. Anyone can do precisely the same thing the poll worker did first thing in the morning: use a portable electronic device to send a challenge to the voting machine over the serial cable and check the response to see if the signed checksum of the software is correct and has a valid signature (A technically challenged voter could bring a tech-savvy friend to verify the machine for him or her). Since some voters will not wish to take part in machine verification, the user interface should make it easy to bypass this step if desired, to allow the voter to immediately begin the voting process.

Attestation's benefits are the protection of voter privacy (successful attestation under our assumptions can show that the voting machine software did not record voter information) and making it more difficult to compromise a voting machine. Because the source code is public, a voter can now have more confidence that the machine is functioning correctly. In the past simple software modifications could have violated voter privacy or mis-recorded votes. Now, an attacker must violate one of our attestation assumptions to run malicious code on an attested voting machine. If verification succeeds while violating our attestation assumptions, election integrity is not compromised, because the voting machine will issue human-verifiable paper ballots that can be easily checked by the voter.

The on-screen directions tell the voter to swipe the voting token with the reader as shown in Fig. 3. The computer then looks up the voter record for VID_i , computes hash $(S_{i1} || S_{i2})$, and compares it to the value stored in the record. A match means two things. First, the voter got the card at home (assuming no one intercepted S_{i1} from the registrar to the post office to the voter), thus at least has access to the mailbox at the address given at registration time (to get S_{i1}). Second, that the poll worker authenticated the voter and gave S_{i2} (i.e., voter did not just sneak in the back door).

Next, the voter is asked to enter his password. The hashed password value is compared to the stored hashed value in the voter's record. If they match, the voter is approved and may vote. If they do not match, the voter can try again up to k times before being locked out. In addition to S_{i1} and S_{i2} (something you have), the password (something you know) is the second line of defense. Without all three values (S_{i1} , S_{i2} , and the password), no one can vote electronically and must use a provisional ballot. This is the only part of the voting system that *requires* a change in the voting process, but a password that the voter may write down should not pose a large difficulty.

Now the voter is presented with the various races

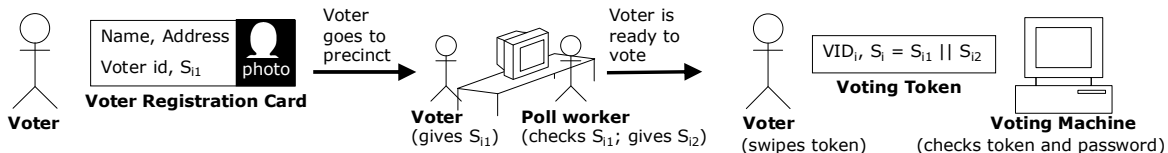


Fig. 3. The voter turns in the card mailed to him for a voting token and then uses the token to vote.

(President, Governor, etc.) one at a time and is given the opportunity to select a candidate for each one. Here is where the multilingual, large font, audio, and other capabilities of the machine shine. At the end, the machine displays a screen showing all the choices and asks if they are correct. If not, the voter can make changes, thus avoiding all the problems seen in the 2008 Minnesota Senate election [41]. If the voter confirms that the choices are correctly recorded, the vote is signed with an auto-generated signing key, encrypted with the precinct public key, and recorded on the storage medium (e.g., CD-ROM or flash memory), and the smart card is overwritten to prevent reuse. There should be a table with initially blank vote slots on the recording medium and one chosen at random (using random numbers from the TPM) to prevent officials from determining after the election how the k th voter voted by examining slot k on the output medium. If a CD-ROM is used, this might require reprogramming the firmware slightly.

A subtle attack exists at this point. Unlike most voting systems, the poll workers are not fully trusted in this design. If a poll worker were to create n identical smart cards (e.g., containing his own information), then the poll worker could use each card to cast a vote during election day. Our defense is simple. The voting machine will also record (to a random location) a hash of the voter's secret, S_i . By having each voting machine check this hash before casting a vote, this limits someone to casting v votes for v different voting machines. Operational procedures that prohibit people from carrying more than one smartcard around after the polls open can also help deter this type of attack. In any case, we view this attack to be unlikely. The attacker has several constraints: a poll worker must help (or be the attacker), the votes can only be cast on election day, and the attack must happen while the polling place is open.

To finalize the vote, the machine prints a signed human-verifiable paper ballot for each race. Having a ballot per race protects against an attacker asking a voter to fill out an entire ballot in a specific manner and later show this ballot to the attacker. Most currently deployed voting machines use cheap printers that sometimes jam; we assume that better quality printers are in use (as on ATMs). The voter is instructed to verify the ballot and put it in the ballot box under the watchful gaze of the poll workers. In the event of a disputed election, the paper ballots are optically scanned or counted by hand. These are the real votes. The machine totals are just preliminary tallies to give people a rough score just after the polls close. With signed paper ballots, a machine cannot undetectably change the election results.

In addition, the machine uses a TPM-generated random number to print out a separate piece of paper for each race with the precinct ID and a random value (unique across all the votes in the precinct), the political office, and a URL on it. Ideally, a poll worker physically stamps the paper (a valid receipt must have a stamp and be signed with a key from a

machine in the specific precinct) and the voter is told to take this piece of paper home. However, stamping each piece of paper may not be feasible, and the digital signature should suffice (This assumes hiding a small cryptographic key is easier than stopping robbers of the stamp). The random number is recorded along with the vote.

Step 8: Tabulating the Votes. When the last voter has voted and the doors locked, the head poll worker goes to each machine in turn and enters a secret code to end the election. The machine then signs the stored votes to mark them as complete and also prints out a ticket with the results, all in the presence of citizen and political party observers. When all the votes have been collected, the recording media are put into a briefcase and locked. The ballot box and briefcase are now securely escorted to headquarters. The head poll worker calls up the county on the phone to report the preliminary results. It is not done electronically because that opens up too many new attack scenarios.

Step 9: Publishing the Result. As soon as is practical after the vote-bearing storage media arrive at the county registrar (in the presence of the parties and citizen observers), they are read in on a computer whose open-source software has gone through our verification process. As a check, the process could be repeated on several computers, possibly supplied by different (political) parties, and combined with randomized manual recounts of a small percentage of the ballots. At this point the county will have a list of {random number, political-office, vote} tuples for each cast vote.

Many paper-based schemes have been proposed to allow the voter to check the integrity of the election without being able to sell a vote [7, 8, 14, 35]. These verification designs are clever in their allowing the voter to verify their vote while keeping their vote secret. However, voters and legislators will have significant trouble understanding how verification works and knowing their vote is actually counted. This complexity precludes system adoption. Aside from verification complexity, receipts pose additional difficulties. Although a voter may trust his verifiable receipt, an attacker can still compromise an election in a way that does not break verification. Although voters may have a verifiable receipt, verification presents a new capability for an attacker: forging bogus receipts to try to get the election thrown out. Based on these issues, we present a simple and transparent verification scheme that should be understandable to most voters and politicians.

In our verification design, the county officials can post the entire list of voting tuples onto its website. This protects election integrity by allowing every voter to verify his or her own vote. If county officials cheat and modify 1% of the votes and 1000 voters check their votes, the probability of undetected cheating is then $0.99^{1000} \approx 0.004\%$. While this scheme preserves the voter's privacy (since only the voter

has the random number printed after voting), voter coercion may become a concern (although still easily done with absentee ballots). To combat voter coercion, voters can switch their receipt with someone from a different party and later show the “required” vote (similar to Rivest and Smith [38]). Receipt swapping can be done with either a trusted friend or perhaps through a receipt-swapping website. Because a swap may involve a bogus receipt, both participants should verify signed receipts themselves or use a friend to verify receipts for them. Unlike floating receipts where voters must check someone else’s vote [38], voters maintain the ability to check their own votes. We believe voters will have some motivation to check if their own votes were recorded but very little motivation to check on the vote of some random unknown person.

Our human-readable receipt solution does not solve the problem where challenges to results can erode voter confidence. Performing a recount on a single challenge would be expensive, but ignoring a percentage of them could hurt the public’s trust of the voting process. If receipts are provided in an election, a policy that balances the voters’ trust and the expense of a recount should be established and followed.

A benefit of our design is that people will easily understand the one-to-one mapping of their number (or detailed vote information) to the site. To get a scheme accepted, it is essential that politicians and voters be able to understand it. With this simple design, voters are motivated to check their receipt. If a valid receipt’s vote is displayed, they can assume their vote was counted.

4. Discussion

This voting system allows anyone, in a simple way, to verify the final tally (sum the votes at the receipt’s URI) while providing each voter a way to verify that his own vote was cast for his own candidate. The voter registration changes of using a password and establishing a secret that will be used on election day helps protect the voter’s vote. No one can cast a vote without the necessary voting token and password. Election integrity is preserved by voters looking up their own votes on the election website. The cost of this simplified voting scheme is that vote selling is now possible with both electronic receipts and the much simpler absentee ballot route. As absentee ballots become more common [28], making the in-person voting system more complicated in order to prevent something that can be easily circumvented with an absentee ballot is a poor tradeoff.

Part of the challenge of voter authentication is our use of passwords. In our system, passwords defend against attacks where someone (e.g., a poll worker) records votes for registered voters that do not show up at the polling place and have not voted absentee. There is no way for the poll worker to vote without knowing (or guessing) the password. The voter is already required to have something to vote (S_i), but the password makes the voter know a secret established at

registration. However, drawbacks exist such as voters forgetting passwords that will increase the number of provisional ballots used (we initially expect this). If no voting authentication mechanism is in place, poll workers can change election outcomes simply by voting in the place of registered voters that do not show up at the polling place (assuming the poll workers are able to get S_i). Using passwords helps thwart these damaging attacks. Other voting schemes are vulnerable to these attacks.

Related to the problem of verification is the trust required for inserting keys into a machine. Using a cryptographic key in a voting machine will require trusting the hardware. Our solution uses a TPM for its hardware protection. If the TPM’s endorsement key (EK) were revealed by a malicious hardware manufacturer, this would undermine the election integrity. However, this compromise requires a change in the manufacturing process (we assume that the EK is generated inside the TPM). For the precinct keys, our main defense is splitting keys after their generation and not rejoining the keys until election day. One must break the machine’s hardware protection to retrieve the key after its machine insertion.

One of our main goals was simplicity in the voting system. Accordingly, the only additional burden on the voter is the requirement of using a password, a concept most voters are already familiar with in other contexts (voter passcodes have been used in a recent election in Hawaii [18]). Although attestation is unusual, it is conceptually simple (“Is the right software running on the machine?”) and is optional. Furthermore, to make attestation practical and simple, voters can use easy-to-use smart phone software to download the necessary data from a website they trust. The software can do all of the checking and warn the voter if anything is amiss.

Issuing voter receipts is one area that also needs attention. Receipts hold great promise, but they need to be carefully tested before deployment. In our view, the main purpose of the receipt is so the voter can see that his or her vote was counted correctly. As a by-product, the integrity of the election is strengthened as each voter verifies his own vote. The main issue is in all the ways the verification process can be abused. We believe reliable verification to be an open problem. We have voting receipts in our system, but many attacks, including forged receipts, remain. Someone could make a fake receipt. Election officials may not know if a machine malfunctioned, or if the voter is cheating. As a last defense against forged receipts, a random paper trail audit that identifies legitimate receipts (e.g., a receipt must share the same ballot number as found on the paper trail) could reveal possible malfeasance. We will investigate these issues in future work.

5. Related Work

Karlof et al. conducted a systems analysis [21] of Chaum’s visual cryptography receipt scheme [6] and Neff’s VoteHere [30] scheme. Although this work was primarily on

the system implementation of cryptographic voting protocols, they showed many different areas of weaknesses in these voting systems including subliminal channels, social engineering, denial of service, and other human factors. Our work also concentrates on the systems aspects of a voting system, but our voting system provides election integrity using simpler methods.

Some of the main functionality of an electronic voting system may be entrusted to a machine including printing a ballot, validating a ballot, and storing cryptographic keys. To protect against machine threats, other types of voting systems advocate the use of trusted hardware [10, 21, 39], and some suggest verification of software integrity [13]. Because a trusted platform module (TPM) is a hardware device that can be used to store secrets, we use this device in our voting system to make an attack on voter privacy and forged ballots more difficult than an attack on an unprotected machine.

Our attestation approach is inspired by Kauer [22] who first created an authenticated boot loader using skinit on an x86 AMD processor. Because the hardware and software of a voting machine is known, once we get a machine into a known state, we can similarly verify a meaningful configuration. Later work by McCune et al. discuss applications using the TPM to protect data (sealed storage) that can be combined with the dynamic root of trust [27]. In our implementation, we plan to experiment with sealed storage to disallow execution of the voting software unless machine-based attestation is successful. Although some voters may still want to attest the machine, this could provide additional assurance about the machine's configuration even when voters choose not to attest.

The OVC voting system [23] is similar to our work in its use of open-source code, but there are many differences. In the OVC system, the voter has the option to verify the ballot by using another verification machine in the polling station. Unfortunately, there is no possibility for a voter to verify his vote actually counted. This system is incompatible with our goal of protecting election integrity and allowing individual voter verification.

6. Conclusion

The procedures and techniques described above using open-source software and shared keys provide a basis for elections that people can have confidence in and which are much harder to tamper with. In particular, the entire system has to be made secure, starting at the top of the election chain. From election key generation to the final count, redundant safeguards are built in at various places to prevent tampering at various places in the process.

By addressing the lessons of past elections with a more auditable registration system and better voter ID cards, election integrity is bolstered. The fully audited registration process helps record each voter's registration. For election day authentication, voters have a relatively strong authentication

token (S_{i1} on a voter ID card). When the voter goes to vote, they now have (S_{i2}) and know (a password) secrets that no one else has in order to vote. After the voter has voted, the voter can use their human-readable verification receipt to confirm their vote was included in the final tally, and statistical paper-based audits provide an additional defense of the reported result's integrity.

Protections that preserve election integrity should help guide designers of voting systems in avoiding potential attacks. We mitigate several attacks with our use of open-source code, through open and public design of the election procedures, and by hardware protection for cryptographic keys. By using these mechanisms to defend election integrity, a system like this may begin to approach a situation in which electronic voting systems can begin to be trusted.

Acknowledgement. We thank Thomas Quillinan, Martijn Warnier, Jeff Napper, Srijith K. Nair, and the anonymous reviewers for their valuable comments.

7. References

- [1] AMD64 Architecture Programmer's Manual. Volume 2: System Programming. AMD Corp., Sept. 2007.
- [2] A. Aviv, et al. Security Evaluation of ES&S Voting Machines and Election Management System. In *Proceedings of the 2008 USENIX/ACCURATE Electronic Voting Technology Workshop*, Jul. 2008.
- [3] J. Benaloh. Administrative and Public Verifiability: Can We Have Both? In *Proceedings of the 2008 USENIX/ACCURATE Electronic Voting Technology Workshop*, Jul. 2008.
- [4] D. Bowen. Secretary of State Debra Bowen Moves to Strengthen Voter Confidence in Election Security Following Top-to-Bottom Review of Voting Systems. Press Release. Aug. 3, 2007.
- [5] K. Butler, et al. Systemic Issues in the Hart InterCivic and Premier Voting Systems: Reflections on Project EVEREST. In *Proceedings of the 2008 USENIX/ACCURATE Electronic Voting Technology Workshop*, Jul. 2008.
- [6] D. Chaum. Secret-ballot Receipts: True Voter-verifiable Elections. In *IEEE Security & Privacy Magazine*, 2(1):38–47, Jan.–Feb. 2004.
- [7] D. Chaum and P. Ryan. A Practical, Voter-Verifiable Election Scheme. In *Proceedings of the 10th European Symposium on Research in Computer Security*. Sept. 2005.
- [8] D. Chaum, et al. Scantegrity II: End-to-End Verifiability for Optical Scan Election Systems using Invisible Ink Confirmation Codes. In *Proceedings of the 2008 USENIX/ACCURATE Electronic Voting Technology Workshop*, Jul. 2008.
- [9] H. Clark. Democratic Primary: Voter Lists Called into Question. *Santa Fe New Mexican*, Feb. 25, 2008.
- [10] M. Clarkson, S. Chong, A. Myers. Civitas: Toward a Secure Voting System. In *Proceedings of the 28th IEEE Symposium on Security and Privacy*, May 2008.

- [11] M. Doig. Analysis Points to Bad Ballot Design. Herald Tribune. Dec. 5, 2006.
- [12] A Master List of 70+ Voting Machine Failures and Miscounts by State. <http://www.commoncause.org/VotingMachine-FailuresMasterList>.
- [13] A. Feldman, J. Halderman, and E. Felten. Security Analysis of the Diebold Accuvote-TS Voting Machine. In *Proceedings of the 2007 USENIX/ACCURATE Electronic Voting Technology Workshop*, Aug. 2007.
- [14] K. Fisher, R. Carback, and A. Sherman. Punchscan: Introduction and System Definition. In *Proceedings of the 2006 Workshop on Trustworthy Elections (WOTE)*, Jun. 2006.
- [15] L. Gong, M. Lomas, R. Needham, and J. Saltzer. Protecting Poorly Chosen Secrets from Guessing Attacks. In *IEEE Journal on Selected Areas in Communications*, 11(5):648-656, June 1993.
- [16] R. Gonggrijp and W. Hengeveld. Stichting “Wij vertrouwen stemcomputers niet.” In *Proceedings of the 2007 USENIX/ACCURATE Electronic Voting Technology Workshop*, Aug. 2007.
- [17] J. Hall. Transparency and Access to Source Code in Electronic Voting. In *Proceedings of the 2006 USENIX/ACCURATE Electronic Voting Technology Workshop*, Aug. 2006.
- [18] Honolulu Hosts Nation’s First All-Digital Election. Associated Press. May 24, 2009.
- [19] Intel 64 and IA-32 Architectures Software Developer’s Manual Volume 2B: Instruction Set Ref., N-Z. March 2009.
- [20] D. Jones. Misassessment of Security in Computer-Based Election Systems. In *RSA Lab Cryptobytes*, 7, 2 (Fall 2004) 9-13. http://www.rsasecurity.com/rsalabs/cryptobytes/CryptoBytes_Fall2004.pdf.
- [21] C. Karlof, N. Sastry, and D. Wagner. Cryptographic Voting Protocols: A Systems Perspective. In the 14th USENIX Security Symposium, August 2005.
- [22] B. Kauer. OSLO: Improving the Security of Trusted Computing. In *Proceedings of the 16th USENIX Security Symposium*, Aug. 2007.
- [23] A. Keller, et al. A PC-Based Open-Source Voting Machine with an Accessible Voter-Verifiable Paper Ballot. In *Proceedings of the 2005 Free and Open Source Software (FREENIX) Annual Technical Conference*, Aug. 2007.
- [24] A. Kiayias, et al. Tampering with Special Purpose Computing Devices: A Case Study in Optical Scan E-Voting. In *Proceedings of the 23rd Annual Computer Security Applications Conference (ACSAC)*, Dec. 2007.
- [25] T. Kohno, A. Stubblefield, A. Rubin, D. Wallach. Analysis of an Electronic Voting System. In the *Proceedings of the 25th IEEE Symposium on Security and Privacy*, May 2004.
- [26] S. Majors. Voting Machine Maker Discloses Program Error. Associated Press, Aug. 21, 2008.
- [27] J. McCune, B. Parno, A. Perrig, M. Reiter, and H. Isozaki. Flicker: An Execution Infrastructure for TCB Minimization. In *Proceedings of the ACM European Conference on Computer Systems (EUROSYS)*, Apr. 2008.
- [28] M. McDonald. (Nearly) Final 2008 Early Voting Statistics. Jan. 11, 2009. http://elections.gmu.edu/Early_Voting_2008_Final.html.
- [29] L. Minnite. An Analysis of Voter Fraud in the United States. Demos. Nov. 19, 2007.
- [30] C. Neff. Practical High Certainty Intent Verification for Encrypted Votes. <http://www.votehere.net/old/vhti/-documentation/vsv-2.0.3638.pdf>.
- [31] L. Norden and J. Allen. Final Report 2008-2009 Ohio Election Summit and Conference. Apr. 8, 2009. <http://www.brennancenter.org/page/-/publications/Ohio.Final.Report.pdf>.
- [32] N. Paul and A. Tanenbaum. Trustworthy Voting: From Machine to System. *IEEE Computer*, pp. 23-29, May 2009.
- [33] R. Pierre. Botched Name Purge Denied Some the Right to Vote. *The Washington Post*, May 31, 2001.
- [34] W. Pieters. La Volonté Machinale. PhD Thesis. Radboud Universiteit Nijmegen, 2007.
- [35] S. Popoveniuc and B. Hosp. An Introduction to Punchscan. In *Proceedings of the 2006 Workshop on Trustworthy Elections (WOTE)*, Jun. 2006.
- [36] E. Proebstel, et al. An Analysis of the Hart Intercivic DAU eSlate. In *Proceedings of the 2007 USENIX/ACCURATE Electronic Voting Technology Workshop*, Aug. 2007.
- [37] R. Rivest and J. Wack. On the Notion of “Software Independence” in Voting Systems. Draft version, July 28, 2006. <http://vote.nist.gov/SI-in-voting.pdf>.
- [38] R. Rivest and W. Smith. Three Voting Protocols: ThreeBallot, VAV, and Twin. In *Proceedings of the 2007 USENIX/ACCURATE Electronic Voting Technology Workshop*, Aug. 2007.
- [39] D. Sandler and D. Wallach. Casting Votes in the Auditorium. In *Proceedings of the 2007 USENIX/ACCURATE Electronic Voting Technology Workshop*, Aug. 2007.
- [40] C. Thompson. Can You Count on Voting Machines? *New York Times*, Jan. 6, 2008.
- [41] T. Bibbetts and S. Mullis. Challenged Ballots: You be the Judge – Round 1. Dec. 1, 2008. http://minnesota.publicradio.org/features/2008/11/19_c_hallenged_ballots/round1/.
- [42] I. Urbina. States’ Actions to Block Voters Appear Illegal. *New York Times*, Oct. 8, 2008.
- [43] A. Yasinsac, et al. Software Review and Security Analysis of the ES&S iVotronic 8.0.1.2 Voting Machine Firmware. Feb. 2007. <http://www.cs.berkeley.edu/~daw/papers/-sarasota07.pdf>.