Trustworthy Voting: From Machine to System

Nathanael Paul and Andrew S. Tanenbaum
Vrije Universiteit, Amsterdam

The authors describe an electronic voting approach that takes a system view, incorporating a trustworthy process based on open source software, simplified procedures, and built-in redundant safeguards that prevent tampering.

After the voting debacle in the Florida presidential election of 2000, many jurisdictions turned to electronic voting machines that were not much more than PCs with touch screens. These machines were as problematic as punch-card systems, plus they made recounts impossible—a complication that drove many jurisdictions back to paper ballots.

But a return to marking choices on paper is also a return to the problems that prompted the use of electronic machines;[1] it is essentially a step backward. Electronic voting has real advantages over paper ballots as long as the focus is on a voting *system*, not a voting *machine*. Rather than concentrating solely on more advanced cryptographic algorithms, designers should be viewing the problem from a system perspective, considering all the pieces and striving for defense in depth. At each design step, they should anticipate attacks at a level that borders on paranoia.

With these aims in mind, Vrije Universiteit has devised an electronic voting system that is both practical and resistant to tampering. We are currently implementing the electronic voting machine software and intend to make the source code freely available this year.

SYSTEM GOALS

Electronic voting offers myriad benefits—from multilingual operation to the prevention of overvoting—but to be trustworthy, a voting system must satisfy three main goals:

- ensure the election's integrity,
- allow results to be audited, and
- be sufficiently understandable that voters and politicians will have confidence in using it.

The election process involves diverse groups, each with sufficient motive and opportunity to influence results. The Secretary of State, who runs the election, is a partisan elected official who (secretly, but fervently) hopes that his party's candidate will win. A partisan county registrar can alter registration data to cause problems on election day; the voting machine's manufacturer might include software to cast, say, every 30th vote for its favorite candidate. The compromise would be large enough to throw a close election but small enough to put the results within the exit polls' margin of error.

The system must allow audits because, if there is a dispute,[2] a recount is mandatory. Requesting a machine to reread the result is pointless because it will merely read out the initial result.

Finally, the voters and the politicians must have confidence in the system. A prerequisite to that confidence is the ability to understand how the system works. Many papers on voting systems describe cryptographic techniques, but cryptography alone does not build confidence in voters. Cryptography is only one method for achieving trustworthiness, and designers should view it as but one aspect of a larger system.

A NINE-STEP PROCESS

Our voting system adds transparent operational procedures and open-source code to standard, well-tested,

SOFTWARE ATTESTATION OF THE VOTING MACHINE

The Trusted Platform Module (TPM) lets a poll worker or voter verify in real time that the voting machine is running the open source software that it is supposed to be running. Central to that verification ability is the skinit instruction. Figure A shows how the skinit instruction helps complete the five functions needed for attestation:

- Disable interrupts, direct memory access to a 64-Kbyte region of memory, and paging.
- Verify that all cores but the one running skinit are disabled.
- Run a hash on the contents of the 64-Kbyte memory region.
- Store the hash in a specific TPM register.
- Execute code stored in the 64-Kbyte region of protected memory.

Anyone can attest a voting machine's software by asking for a TPM-signed hash of the software (an X.509 certificate for the corresponding public key is also available, so anyone can verify the hash). If the signed hash matches the published hash of the open source code, the individual requesting verification can be confident that the machine's code has not been altered.

However, confidence in the attestation algorithm's result is only one piece of the attestation puzzle. The hardware must also be correctly functioning and uncompromised, and the signing key must not have been leaked. Attestation can be successful even if neither of these conditions holds, and the resulting violations are difficult to detect. An incorrectly operating machine might record the voter's choices, for example, and a leaked key could open the opportunity to forge a completed ballot. Thus, to preserve election integrity even under such conditions, our scheme uses human-verifiable paper ballots and paper receipts.

Hence, if only the TPM has the private key and the signed hash of the 64-Kbyte memory content is correct, the 64-Kbyte program stored in that memory, or *checker*, also must be correct. The checker hashes all the memory content (including the operating system), the CD-ROM, main BIOS, CD-ROM BIOS, and any other BIOS present. It also keeps interrupts and DMA disabled so that the attested code never loses control. Once verified, it always remains valid, since the machine is not on any network. Once attestation is complete, the rest of the process is technically straightforward.

Four parts of our voting system use the skinit instruction to attest that the checker is correct. If the checker is known to be correct and verifies that the rest of the memory content is what it should be, attestation is complete.

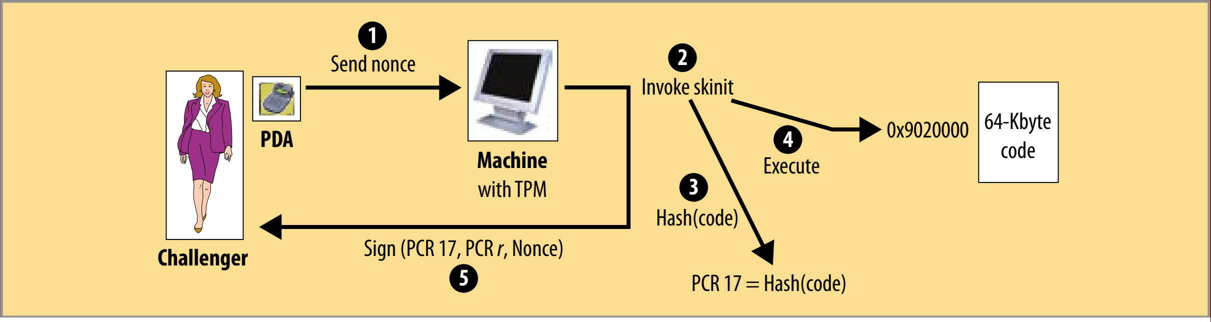


Figure A. Attestation using the skinit instruction. (1) The algorithm accepts a random value (a nonce) as input and (2) executes skinit, which in turn executes steps 3 and 4. Once invoked, skinit disables paging, stops DMA to the memory containing the checker, disables interrupts, and verifies single-core execution. (3) skinit then computes and stores a hash of the checker program in the TPM platform configuration register (PCR 17), and (4) executes the checker, which writes its result (the hash of the machine's data and software) into a different PCR register, *r*. (5) Once skinit is finished, the machine returns the TPM signature of {PCR *r*, PCR 17, nonce}. If this value is correct, the attested software and data have not been altered.

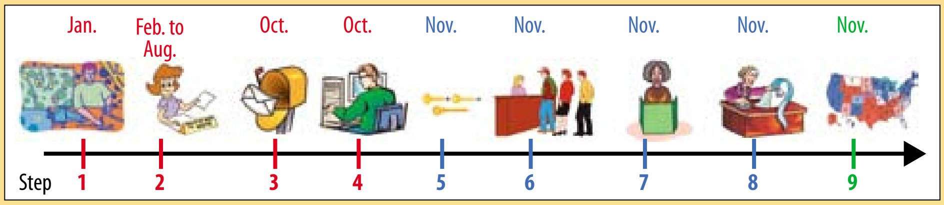
cryptography. As Figure 1 shows, the underlying voting process has nine steps that can take place up to a year before election day. For simplicity, we use US government terms to explain these steps, but the process is suitable for other democratic governments.

To protect election keys and voter privacy, the system uses open source voting software and lets anyone verify that the published software is indeed running on the voting machine. *Attestation*—the process of verifying that the software now running is the published software—is the main technical challenge. There is also the challenge of getting states to use open source software, but that is a political and legal issue.

In our scheme, the system performs attestation by computing the hash—a cryptographic checksum—of the published (executable) software and the running software. The user can compare the two hashes. For hash algorithms such as SHA-256, it would be extremely difficult to create malicious code whose hash matches the published voting software's hash.

The hard part is computing the hash over the machine's software in a way that can be trusted. To do this, we use the Trusted Platform Module (TPM), a device that is already part of many modern PCs. AMD processors that support version 1.2 TPM chips have a special instruction, called skinit, that can be used for software attestation (Intel's GETSEC[SENTER] instruction is similar). The "Software Attestation of the Voting Machine" sidebar describes using the skinit instruction's role in attestation.

Step 1: Generate and distribute precinct master keys
Step 2: Create voter registration records
Step 3: Mail proof of registration to voters
Step 4: Prepare voting machines
Step 5: Assemble key pairs at precincts
Step 6: Check in voters
Step 7: Have voters cast their votes
Step 8: Tabulate votes
Step 9: Publish results



🟥 Pre-election day 🟦 Election day 🟩 Post-election day

Figure 1. Timeline of the nine-step electronic voting process. The process can start up to a year before the election, with steps 1 through 4 extending to election day. Careful preparation enables steps 5 through 9 to take place in rapid succession. The use of open source software, transparent procedures, and simple cryptography gives voters high confidence in the system's trustworthiness.

Step 1: Generate and distribute precinct master keys

Parts of the system require cryptography to ensure secrecy and prevent tampering. Computational load is not an issue, since voting machines don't have throughput concerns, so our system uses public-key cryptography to simplify key management. All public-key systems involve the use of a public (encryption) key and a private (decryption) key. We use three types of key pairs to encrypt and sign voting data:

- key pair 1: Encrypting-decrypting key pair for files on voting and poll worker machines (per precinct),
- key pair 2: Ballot-signing key pair (per voter), and
- key pair 3: Software attestation signing key pair (optionally created per voter).

Procedure. Counties are divided into election precincts, each with a single polling place, typically a school or firehouse. Each precinct (polling place) requires a single public-private key pair (key pair 1) to lock or unlock files on all its voting and poll workers' machines. California, for example, has 24,000 precincts, requiring the generation, storage, and distribution of 24,000 public-private key pairs. Done once per election, these activities are not overly burdensome, since generating 24,000 precinct key pairs can take at most a few hours. The TPM uniquely generates the other keys on the fly when signing a ballot or attestation result, and they are trusted because the TPM has signed them. The TPM's unique and freshly generated signing key is itself signed by a TPM *endorsement key* (EK). The EK is the most trusted key in the TPM and is typically generated when the chip is manufactured, or alternatively, remade per election.

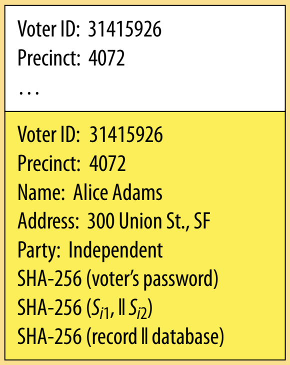
The generation process for the precinct key pairs (key pair 1) starts with the Secretary of State, who sends out a request for bid to vendors of computers that support TPM 1.2. Once the contract is awarded and the machines are in place, the Secretary of State invites all the political parties and the media to a public-key generation event—as early as a year before the election. Each party can send one party officer and one technical expert that the party chooses to inspect the machine's endorsement key certificate. The certificate, which typically comes from the TPM manufacturer, verifies that a machine has a legitimate TPM. The machine will use the EK to sign an attestation identity key. The machine will then use the identity key to sign the software hashes stored in TPM registers.

In addition to the EK certificate, the expert can use a platform certificate that a third party has signed to verify that the TPM conforms to specifications. These steps assume that people trust the certificate authorities and that reliable certification processes are in place. The group then performs attestation on the machine's software before running the key-generation software.

After attestation, the verified software generates the public-private key pairs for the election, and the machine signs the new public keys and stores them on a CD. Because the group has just checked the machine's integrity, the keys are trusted, but the group can perform additional checks to ensure the keys' integrity.

The key-generation software splits each private key into two or three parts using any proven secret-sharing scheme.[3] The scheme must ensure that, without all the parts, the secret is not recoverable. This step is analogous to banks not giving the full vault combination to any one officer so that opening the vault requires more than one person.

The key-generation software writes each part of each secret onto some tangible medium such as a contactful smart card (that emits no radio signals). Smart cards with RS-232C serial line interfaces make code verification easier. These cards have extremely simple device driv-


Voter ID: 31415926
Precinct: 4072
…

Voter ID: 31415926
Precinct: 4072
Name: Alice Adams
Address: 300 Union St., SF
Party: Independent
SHA-256 (voter's password)
SHA-256 (S_{i1} || S_{i2})
SHA-256 (record || database)

Figure 2. A record in the voter registration file for precinct 4072. At the top is a plaintext voter ID and precinct number. The registrar's machine encrypts the rest of the record using the precinct's (public) key.

ers relative to USB drivers, which are insanely complex. Because the PC can have a PCI board with a dozen serial lines, the software can write on many smart cards in parallel. If necessary, the group can vet a second or third PC and use it in a similar way, enabling the production of all the smart cards in one day, while the political parties' technical experts scrutinize the PCs and each other.

Example. As a practical example, assume that each private key has two parts. Once the software finishes writing on all the smart cards for a particular county, the officials put all the part A's into a briefcase, which the county's registrar of voters takes back to his county. All the part B's go into another briefcase, which the county sheriff takes back separately to the same county. The two lock their respective briefcases in different buildings until the election. In a more paranoiac scenario, each key could have four parts, and a briefcase of respective parts would go to representatives of the two leading political parties in each county, as well as to the voting registrar and sheriff. Consequently, it should be nearly impossible for anyone to assemble the key before election day.

Our generation and distribution scheme assumes that no one loses any part of any key and that no key holders collude. More flexible threshold schemes that allow full key recovery using a subset of the pieces are well known and might also be suitable.[3]

Step 2: Create voter registration records

Once the Secretary of State has distributed all the keys, voter registration can begin. To register, a voter goes to the county office with whatever else the law requires (for example, proof of residence). The registrar uses this information to insert a newly created voter record into an append-only database. An option is to include a digital photo of the voter in the record and print it on the card. Figure 2 shows a sample record.

Part of the record contains the hash of the voter's password. Because voters might not trust county officials, they can bring a notebook or cell phone preloaded with their password. To illustrate, suppose Mr. Jones has preloaded his password on his cell phone. From his cell, Mr. Jones sends the SHA-256 hash of his password to the registrar's computer using a serial or USB cable. Mr. Jones has used his own device to deliver only the hash of his password, *not the password itself*, thereby foiling any county employees who might want to vote as Mr. Jones. Voters without a mobile device must use the county's computer to enter their password and trust that the county won't steal it.

The registrar's computer also generates a secret for voter i, S_i and breaks it into two parts: S_{i1} and S_{i2}, where $S_i = S_{i1} \| S_{i2}$ (|| being concatenation or XOR), computes SHA-256($S_{i1} \| S_{i2}$), and stores this in the voter's record. Each part also is encrypted with a separate county public key to be used later in Step 4 to prepare voting files.

After creating each voter record, the software immediately cryptographically hashes the record along with the rest of the voter database and encrypts the record with the public key of the voter's precinct to prevent tampering (this also protects against brute-force attacks on the hashed passwords). The record is exposed for only a few seconds and, once encrypted, is safe from any future tampering, since our scheme makes it nearly impossible to assemble the precinct's private key without the parts.

Step 3: Mail proof of registration to voters

A few weeks before election day, counties in many states mail each voter a sample ballot and booklet with the candidates' statements, information about ballot initiatives, and so on. In our scheme, that package also contains a single-use card printed on security paper or containing a chip—anything hard to forge. The card serves as proof of citizenship, residence, and registration so that the voter is not burdened with supplying proof at the polls. The card is free, as is the sample ballot.

More important, the card also contains part of the secret (S_{i1}) generated at registration. It could be printed on the card as characters or as a bar code, or it could be on a chip. The card also contains the address of the polling place, the hours it is open, the voter's digital photo (if taken at registration), and a reminder to voters to bring their password.

Step 4: Prepare voting machines

Each voting machine at a polling place contains a file of all voter records for that precinct, enabling a voter to pick any voting machine. A voter who goes to the wrong polling place will have to cast a provisional (paper) ballot, since the voting machines at that polling place will not have his record.

Once the county registrar and other officials prepare the precinct voting list, they cryptographically hash and

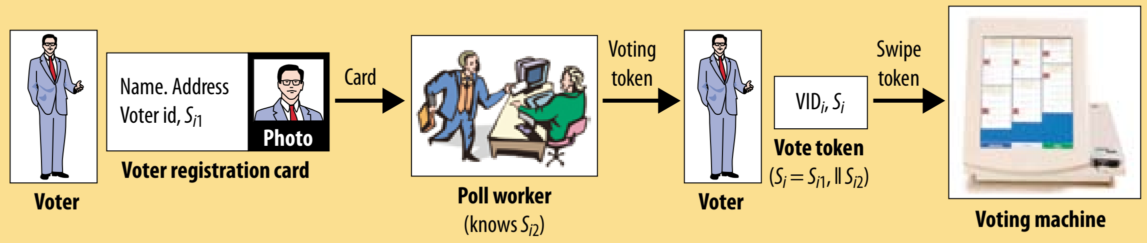


Figure 3. Voter check in. Voters use a card they received in the mail that contains their voter ID and part of the secret generated during registration (S_{i1}). The poll worker retrieves the other half of the secret (S_{i2}) and uses his machine to concatenate the two (S_i) to create a voting token that contains the voter's ID number and S_i. The voter uses the token to access a voting machine.

encrypt it with the precinct's public key and store it on a read-only medium, such as a CD-ROM, which a poll worker can use to boot the precinct's voting machines. The point of hashing and encrypting the entire voter file is to prevent anyone from tampering with it while it is in storage or in transit to its precinct.

To finish preparation, the voting officials create a second CD-ROM for the poll workers' machines. This CD-ROM contains the file with each voter's ID number, name, address, and S_{i2} value (the second part of the secret). The registrar and officials also hash and encrypt this file with the precinct's public key to prevent tampering in storage or transit.

Step 5: Assemble key pairs at precincts

A few weeks before the election, the county posts the EK and platform certificates for each voting machine and the precinct's public key on the county's website. About 30 minutes before each precinct opens for voting, the head poll worker delivers the county's half of the private key, which she obtained from the county registrar, who retrieved his part of the key from its safe. A sheriff's deputy arrives at approximately the same time with the other half of the private key. (If the private key has been divided in more parts, any other part holders also arrive at this time.) Legal sanctions should be in place to encourage showing up on time to prevent denial-of-service attacks aimed at shutting down the polling place. Alternatively, officials can decide to use a threshold cryptographic scheme.[3]

Before booting the voting machines, poll workers inspect them for evidence of tampering. They then boot the diskless machines using the precinct's CD-ROM. An alternative is to vet the machines before the election and hermetically seal them in a way that would make tampering obvious.

Once the operating system and voting software are loaded, the poll workers use a PDA to verify the software using attestation. Because the verified software disables interrupts and DMA, unverified software never gains control. Without a network, the attested code can continue execution without interference.

After poll workers verify the machine's code integrity, they successively insert smart cards with the precinct's private key parts, thereby assembling the final precinct key to unlock the voter file on the CD-ROM. To aid the poll workers, the machine prompts for card insertions. Key assembly can take place outside the TPM because all code on the machine is verified. Because the full precinct key has not been available in one place until that moment, no one could have tampered with the encrypted voter file during transport or storage.

The last task is to verify the poll workers' machines using the same process used to verify the voting machines. The precinct is now ready to accept voters.

Step 6: Check in voters

Figure 3 shows the voter check-in process. The voter goes to a poll worker and hands over the card he was mailed (or gets a provisional paper ballot if he has no card). The poll worker enters the voter's ID in the poll worker's computer, thus bringing up the voter's record, which is now decrypted. The poll worker checks if the name and address (and optionally, a photo) on the screen match those on the card.

The poll worker then asks the voter if he remembers the password he entered during registration. If he does not, he receives a provisional paper ballot. Otherwise, the poll worker scans and enters the voter's S_{i1} value from the card, and the computer concatenates this newly read 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 and reencrypts the voter's record. The poll worker hands the voter the voting token and tells him to go to any voting machine and follow the onscreen directions.

Step 7: Have voters cast their votes

Before casting a vote, the voter might want to verify that the voting machine is indeed running the open source software published on the county registrar's website, although this is completely optional. Anyone can do precisely the same thing the poll worker did that morning: Use a portable device to send a challenge to the voting machine over the serial cable and check the response to see if the cryptographically signed checksum of the software is correct and has a valid signature. Again, if the signature is correct, it must have been signed by code running inside the TPM, since the operating system does not have access to the machine's private key and thus cannot forge a correct response.

The onscreen directions tell the voter to insert the voting token into the reader. The computer then looks up the voter record for that ID number (VID_i), computes SHA-256($S_{i1} \| S_{i2}$), and compares that value to the value stored in the record. A match provides two important pieces of information:

The password is the ultimate defense against stolen cards and corrupt county workers.

- The voter has the S_{i1} that was mailed to the address given at registration time.
- The voter didn't just sneak in the back door. He visited the poll worker (to get S_{i2}) and was approved.

The screen then instructs the voter to enter his password, which the machine hashes and compares to the hashed value stored in that voter's record in the voter file. If the values match, the voter can vote. If they do not match, the voter can try his password up to *n* times before the machine locks him out. The password is the ultimate defense against stolen cards and corrupt county workers. It is never recorded anywhere except possibly by the voter, and without it, the voter cannot vote electronically. Consequently, stealing a voter's sample ballot packet and card is pointless because the password is not there.

Once approved, the voter views the ballot and makes the desired selections. All the features of electronic voting are available: multilingual text, large fonts, audio, and so on. At the end, the machine displays a screen of all the choices and asks if they are correct or if the voter wants to change anything. If the voter confirms that the choices are correct, the machine records the vote on some storage medium, such as a CD-ROM or flash memory, and overwrites the smart card with random numbers to prevent its reuse. The recording medium contains a table with vote slots that are initially blank. To keep officials from determining how a particular voter voted, the machine chooses a slot at random using the TPM random number generator. Otherwise, after the election, officials could determine how the *k*th voter voted by examining voting slot *k* in the table.

The voting machine then prints out a human-readable ballot and instructs the voter to verify it and deposit it in the ballot box, which poll workers are monitoring. If the election is later disputed, officials can optically scan these paper ballots or hand-count them. The ballots are the real votes; the computer totals are just preliminary tallies that provide a rough idea of who won soon after the polls close.

After printing the ballot, the machine picks a TPM-generated random number and prints it on a piece of paper along with a URL of a site the voter can later use to check his vote, recording the random number along with the vote. The voter receives separate printouts (one per race), each one with a unique random number, to take home.

Step 8: Tabulate votes

When the last voter has voted, the head poll worker locks the doors and enters a secret code in each machine that signals the end of the election. The machine then encrypts all the results on the storage medium and prints out a ticket with the results—all in the presence of citizen and political party observers. When all the machines have written their votes, the head poll worker puts the recording media into a briefcase and locks it. The worker and an escort take the ballot box and briefcase to headquarters, and the worker phones the county registrar to report the preliminary results. Reporting uses the telephone to avoid new attack opportunities.

Step 9: Publish results

After the box and briefcase arrive at the county registrar's office, where citizen and political party observers have also gathered, the registrar enters the results on a computer running open source software that has gone through attestation. As a check, the registrar could repeat the process on several computers that different parties supply.

At this point, the county has a list of {random number, political-office, vote} tuples for each recorded vote. The county registrar could post the list on the official website so that voters could look up their random number and verify that the machine recorded their vote correctly. This opportunity for checking makes undetected cheating highly unlikely. For example, suppose that county officials attempt to modify 1 percent of the votes. If 1,000 voters check their votes, the probability of undetected cheating is only 0.99^{1000}, or approximately 0.004 percent.

Although our scheme preserves privacy, since only the voter has the random number printed after voting, theoretically a voter could still sell his vote. If the state decides that vote selling is a bigger problem than election tampering, an alternative is to publish the random numbers without the vote. The voter and the vote buyer can thus see that the vote counted, but they cannot see how.

The procedures and techniques we have described work together to yield a trustworthy voting system—one that is secure from the first key's generation to the publishing of results. In our nine-step process, election integrity stems from the voting machine software's open source character, the use of public and transparent procedures, and the voters' ability to personally verify their individual votes. Many proposed schemes let voters check election integrity and prevent them from selling their votes, but all these schemes rely on extremely complicated mechanisms or mathematics that few people understand. When voters ask, "How do I know that voting is honest?" the answer is usually, "Trust us." We anticipate that our system will provide a simpler and more trustworthy alternative.

References

1. T. Tibbetts and S. Mullis, "Challenged Ballots: You Be the Judge," *Minnesota Public Radio*, 3 Dec. 2008; http://minnesota.publicradio.org/features/2008/11/19_challenged_ballots.
2. "A Master List of 70+ Voting Machine Failures and Miscounts by State;" www.commoncause.org/VotingMachineFailuresMasterList.
3. A. Shamir, "How To Share a Secret," *Comm. ACM*, Nov. 1979, pp. 612-613.

***Nathanael Paul** is a research scientist in the Department of Computer Science at Vrije Universiteit, Amsterdam. His research interests are electronic voting, malware, and virtual machines. Paul received a PhD in computer science from the University of Virginia. He is a member of the IEEE and the ACM. Contact him at nate@few.vu.nl.*

***Andrew S. Tanenbaum** is a professor in the Department of Computer Science at Vrije Universiteit, Amsterdam. His research interests are reliable operating systems and security, particularly in ubiquitous systems. He received a PhD in physics from the University of California, Berkeley. He is a Fellow of the IEEE and the ACM and a member of the Royal Netherlands Academy of Arts and Sciences. Contact him at ast@cs.vu.nl.*