CIS 551 / TCOM 401

Computer and Network Security

Spring 2009
Lecture 23

Announcements

- Plan for Today:
  - Grab bag: Homomorphic Encryption / Secret Sharing
  - Analysis of an Electronic Voting System

- Project 4 is due 28 April 2009 at 11:59 pm

- Final exam has been scheduled:
  - Friday, May 8, 2009
  - 9:00am – 11:00am, Moore 216

Blind Signatures

- Digital signature scheme equipped with a commutative *blinding* operation
  - Signer never learns what they signed
  - Like signing an envelope with a window (or with carbon paper)
  - I.e.:   unblind(sign(blind(m))) = sign(m)
- Voting scheme:
  - Voter prepares vote *v*, blinds, and authenticates to Authorization server, and sends vote. Server checks off voter, signs vote, and sends back to voter. Voter unblinds and now has sign(*v*).
  - Voter anonymously sends sign(*v*) to Tabulation server. Server checks signature, then counts vote.

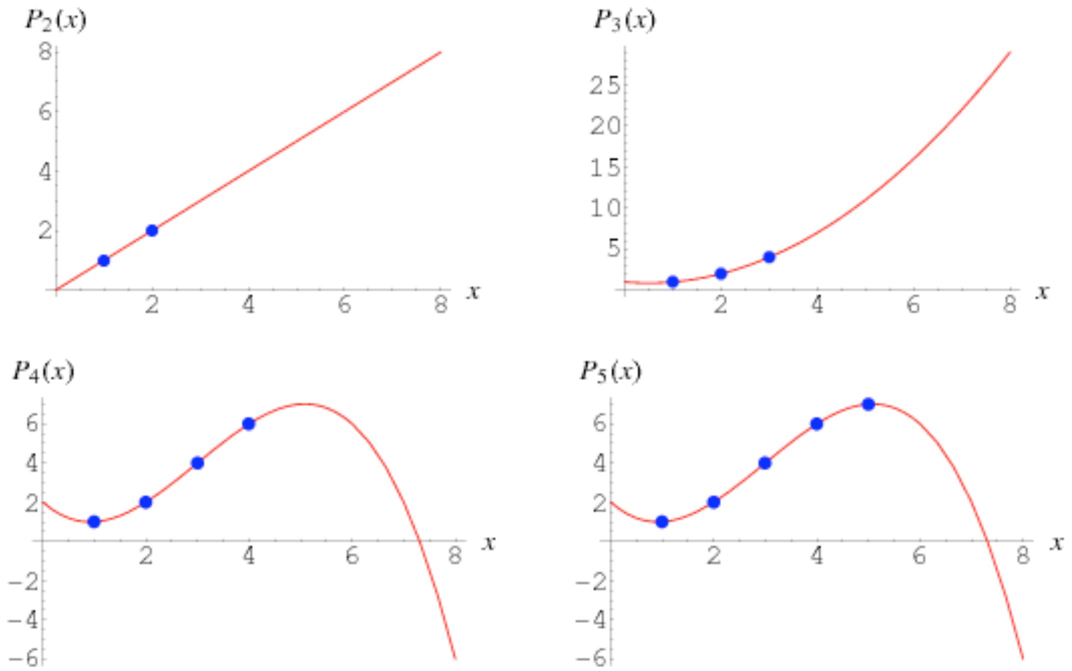
Homomorphic Encryption

- A *homomorphic* encryption scheme has an operator $\star$ such that Enc(m) $\star$ Enc(n) = Enc(m $\star$ n). $\star$ is usually either + or $\times$, never both.
  - E.g. both RSA and El Gamal have $\times$.
- Voting scheme:
  - Suppose scheme has + as homomorphism and votes are either 0 or 1.
  - Voter prepares Enc(0) or Enc(1) as vote, authenticates to Tabulation server, and submits vote.
  - Tabulation server sums all the votes, then decrypts result. Individual votes never decrypted.

Secret Sharing

- How to share a secret among N+1 players:
  - Owner of the secret generates N random bitstrings R1 … RN
  - Player 0 gets $S \oplus R1 \oplus \ldots \oplus RN$
  - Player j > 0 gets Rj
  - All N players can cooperate to recover S -- they just XOR their shares.
- *Threshold* schemes allow k-out-of-N players to recover the secret:
  - Owner of the secret picks a random polynomial f with degree (k-1) such that f(0) = S
  - Player j > 0 gets f(j)
  - If any k players get together, they can use Lagrange interpolation to calculate f(0)
  - If fewer than k players get together, there's no information about f(0).

Lagrange Interpolation



The Lagrange interpolating polynomial is P(x) that passes through n points: $(x_1, y_1 = f(x_1)), \ldots, (x_n, y_n = f(x_n))$

$$P(x) = \frac{(x-x_2)(x-x_3)\cdots(x-x_n)}{(x_1-x_2)(x_1-x_3)\cdots(x_1-x_n)}\,y_1 + \frac{(x-x_1)(x-x_3)\cdots(x-x_n)}{(x_2-x_1)(x_2-x_3)\cdots(x_2-x_n)}\,y_2 + \cdots + \frac{(x-x_1)(x-x_2)\cdots(x-x_{n-1})}{(x_n-x_1)(x_n-x_2)\cdots(x_n-x_{n-1})}\,y_n.$$

Example: 3-out-of-N Secret

- Suppose the secret is S = 7
- I generate (at random) $f(x) = 2x^2 - 3x + 7$
- Then $S = f(0) = 7$
  - Share $s1 = f(1) = 6$
  - Share $s2 = f(2) = 9$
  - Share $s3 = f(3) = 16$
  - Share $s4 = f(4) = 27$

- To recover secret and obtain 3 shares:
  - Example: given s2, s3, s4 = (2,9) (3,16) (4,27)
  - Calculate P(x) as on the previous slide [see blackboard]

Non-Electronic Elections

During the election, voters:

- Go to polling place and give name, address
- Get ballot, enter booth
- Use mechanical punch to punch out perforated holes to indicate vote
- Take ballot cards, put into concealing envelope
- Leave booth, drop envelope into ballot box

After the election:

- Election officials remove ballots from envelopes
- Ballots run through optical scanner to count votes
- Verify by hand:
  - E.g. under California law, 1% of ballots from precincts counted by hand, compared to results from optical scanner

Properties

- Voter must be able to vote
- Votes are secret (no coercion)
- Votes are anonymous
- Voter can verify votes at any point before dropping ballot into ballot box
- Voter can get new ballot any time before placing ballot in ballot box
- Voter votes limited number of times per race, and once per ballot
- Vote tally is accurate and auditable

Disadvantages of paper vs. electronic ballots

- **Low reliability**
  - Many more ways for voters to make errors
  - Paper ballots constitute a single point of failure if lost or damaged; cannot be “copied” without loss of fidelity
  - Just as much software required as in electronic voting (in scanners and printers)
  - Require human judgment to ascertain “voter intent”
  - Paper is essentially an analog medium
- **Poor security**
  - No checksums or encryption possible—data is non-redundant and in cleartext
  - Paper ballots can be manipulated by hand—no tools or skill required
  - Trojan horse argument not avoided--possible in printers, scanners, etc.
- **Excessive costs**
  - Printing
  - Mechanical paper-handling equipment (scanners, printers) required
  - Storage, transport, and storage again
  - Waste

Role of E-Voting System

- Replace manual punch and paper ballots
  - Easier to configure (potentially 100's of ballots in an election)
  - Can handle multiple languages easily
- Replace hand tallying of votes
- Possible more secure

Electronic Voting Requirements

- Must be available
- Must provide simple to use, easy to understand, hard to misuse interface for voter
- Must not be able to associate votes with a particular voter
- Must allow voter to discard votes up to the time the voter officially casts ballot
- Must prevent voter from casting more than limited number of votes per race, or once per ballot
- Voter must be able to verify vote up to time vote is cast
- Must tally votes accurately
- Must provide an *out-of-bands* mechanism for verifying vote tally

Key Ideas

- Separation of Privilege
  - Observers can check everything in paper election
    - Not with e-voting systems to the same degree
- Auditability

Points to Ponder

- What OS does the e-voting system use, if any?
- How long does the e-voting system stay up?
- What is the procedure for getting e-voting machine ready to use?
- How have you tested the user interface?
- How do you handle write-in votes?
- How do you prevent double voting?
- How do you provide anonymity?

Points to Ponder (2)

- How does the system associate votes with voters?
- Does your authentication/ authorization mechanism associate external voter identities with that information?
- What is point at which e-ballot is cast, and voter cannot redo any part of the ballot?
- How do voters change their votes?

Points to Ponder (3)

- How do you check enforcement of limits on voting in a race?
- What support must be provided to ensure that no-one can cast multiple ballots?
- What assumptions does the e-voting system make about procedures and support?
- How can the voter verify that the e-voting system accurately recorded votes cast?
- Does this verification require the intervention of a third party?

Points to Ponder (4)

- What requirements is system designed to meet?
- How do you know it meets them?
- How do you handle updating software, hardware on fielded systems?
- What do maintenance people do when they work with e-voting systems?
- How can you verify systems meets requirements after maintenance, upgrade?
- What audit mechanism, external vote tally, does the system supply and how do you know it is correct?
- How could an auditor use this mechanism to validate results of an election?

Analysis of an Electronic Voting System

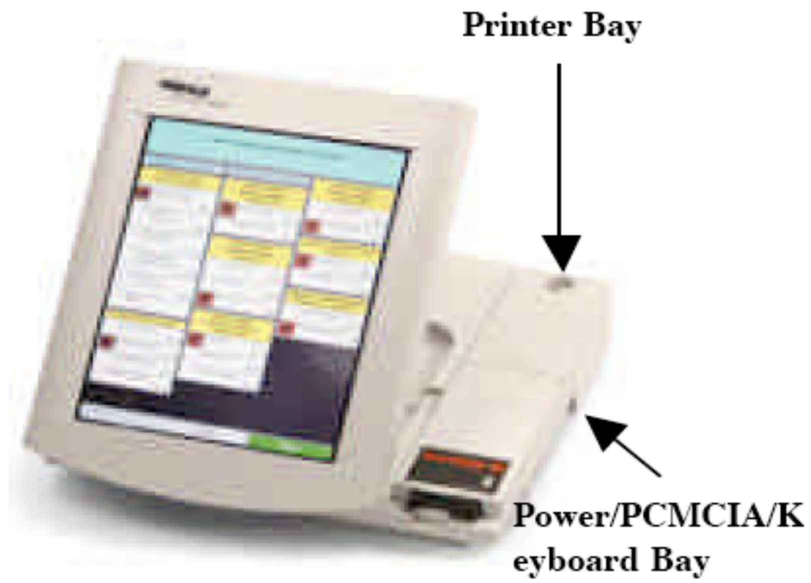
- Paper by: Tadayoshi Kohno, Adam Stubblefield, Aviel D. Rubin, Dan S. Wallach
- 2004 study in IEEE Security & Privacy Conference

- Diebold AccuVote-TS DRE voting system.
- In 2004 Diebold products were used for voting in 37 states.

DRE voting systems

- Direct Recording Electronic voting systems
- Eliminate paper ballot
- Go to polling booth, prove you are allowed to vote
- Obtain smart card and pin
- Enter your voting choice in the voting terminal
- Commit your vote

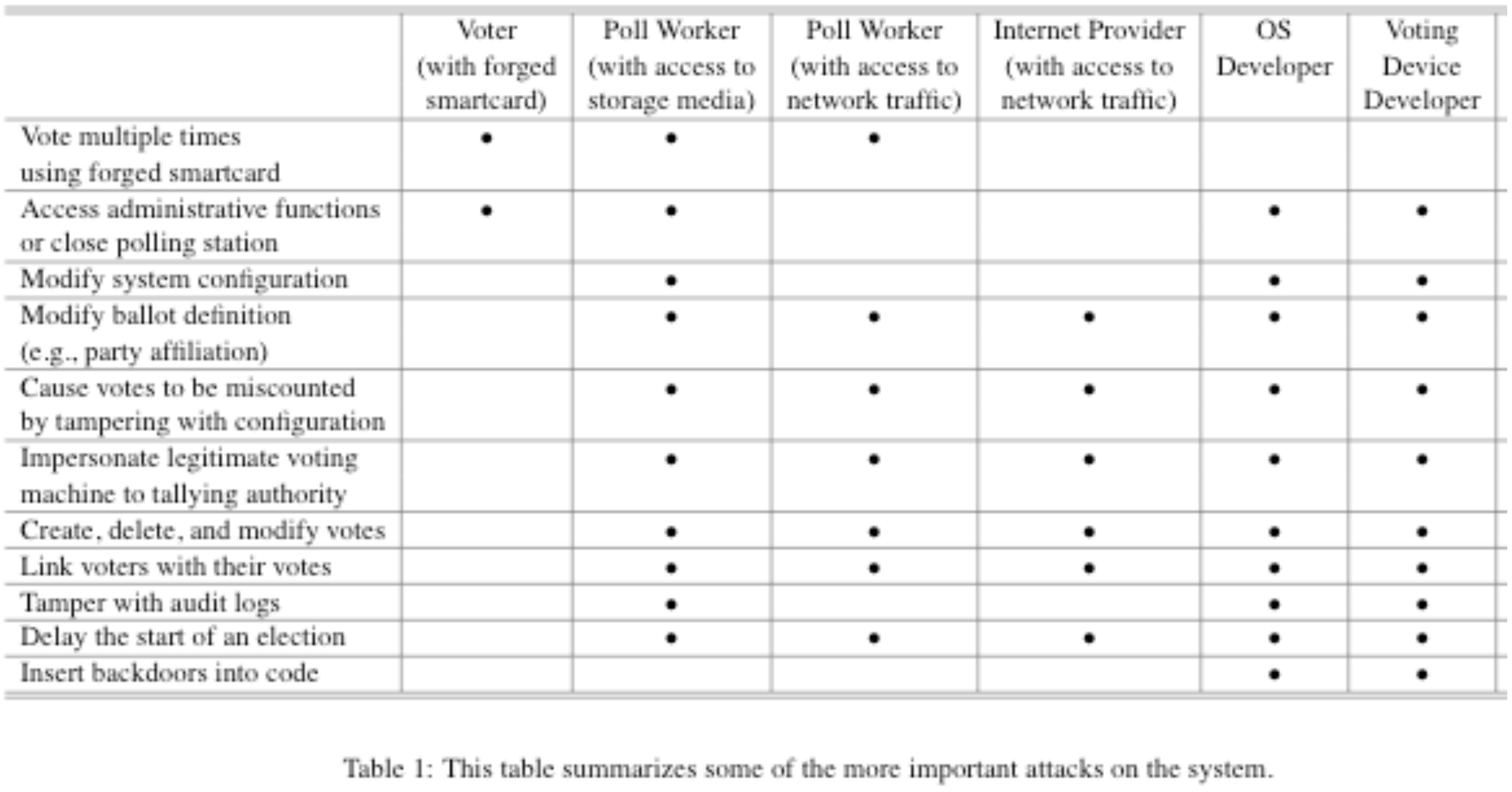
Diebold AccuVote-TS Terminal



Summary

- Voters can use fake smart cards
- Voters can cast multiple votes
- Voters can perform admin actions
  - View partial results
  - Terminate the election process early
- Bad/no cryptography
- C++ and poor engineering practices

Diebold AccuVote-TS 4.3.1



	Voter (with forged smartcard)	Poll Worker (with access to storage media)	Poll Worker (with access to network traffic)	Internet Provider (with access to network traffic)	OS Developer	Voting Device Developer
Vote multiple times using forged smartcard	•	•	•			
Access administrative functions or close polling station	•	•			•	•
Modify system configuration		•			•	•
Modify ballot definition (e.g., party affiliation)		•	•	•	•	•
Cause votes to be miscounted by tampering with configuration		•	•	•	•	•
Impersonate legitimate voting machine to tallying authority		•	•	•	•	•
Create, delete, and modify votes		•	•	•	•	•
Link voters with their votes		•	•	•	•	•
Tamper with audit logs		•			•	•
Delay the start of an election		•	•	•	•	•
Insert backdoors into code					•	•

Table 1: This table summarizes some of the more important attacks on the system.

Code base

- Diebold system leaked
  - CVS repository on an open `ftp` server
  - Commit logs from October 2000 --- April 2002.
  - identified by activist Bev Harris
- C++
  - Buffer overflows in operation
- Windows CE
- Code compiles on regular Windows machines

- 4.3.1 snapshot
- .H
  - 136 files
  - 16414 lines
- .CPP
  - 120 files
  - 33195 lines

System setup

- Ballot definition : election.edb
- Removable media
- Local network
- Dialup internet

Code overview

- Loads the poll information from registry.
- If registry info is missing, then
  - Asks admin to setup the machine
  - Admin
    - COM port for smartcard reader, directories to store files, polling location identifier.
  - Opens election.edb
    - If missing, downloads the file from Internet.

Creating homebrew smartcards

- No crypto operations.
- Thus no authentication of smartcard to terminal.
- Attacker can program smartcards.
- m_ElectionKey, m_DLVersion
  - same for location.
- m_Vcenter
  - learn from good smart cards.
- Wiretap device to understand the protocol?

Casting multiple votes

- Flags: m_CardType

– 0x01	VOTER_CARD
– 0x02	ENDER_CARD
– 0x04	ADMIN_CARD
– 0x08	CANCELLED_CARD

- After voting, reader rewrites card type. Adversary card can just ignore it.
- Or just bring along multiple cards.

Accessing admin functions

- Default Security PINs of 1111 on administrator cards
- Can also just forge the card.
  - We need to learn PIN for the terminal.
  - Know protocol
    - Any PIN works because the system takes PIN from card.
  - Do not know protocol
    - Suppose adversary has a stolen admin card.
    - Snoop on the card traffic and try all 4 byte strings.
- Capability to end polls.

Physical tampering

- Physical access
  - tampering with system configuration is very easy, just change registry entries.
- No protection against physical tampering.
- No encryption/checksum if network used to transfer data.
- Anyone can pretend to be the server/voting machine.
- No sequence numbers
  - deleting records easy.

“Encryption”

- Encryption is used when data is stored on PCMCIA cards sent for counting.
- Impersonating legitimate voting terminals easy. Just change the registry value.
- Data stored on the PCMCIA cards is encrypted with the same hard-coded DES key:

```
#define DESKEY ((des_key*)"F2654hD4")
```

  - Key in use since 1998
- DES CBS mode with an all-zero Initialization Vector, as opposed to random IV required.
- Unkeyed public function (CRC) used for integrity protection

- No authentication of smartcard to voting terminal

Sending votes to server

- Clear text if using Internet. Not sure if they use Internet to send the votes.
- Reorder attack
  - One can reorder ballot order without getting caught. They just store the number of votes for choice 1, 2 …
- Increase number of votes for one candidate, decrease the other.

- DDoS the election server?
- Server IP address was available on the Internet.

Linking votes with voters

- “Randomized serial numbers”
  - Linear congruential generator
  - Bruce Schneier’s Applied Cryptography
- Poll workers can record the order in which people use the machines.
  - Can track the records in the vote database.

Comments in the Source Code

```
// LCG - Linear Conguential Generator
// used to generate ballot serial numbers
// A psuedo-random-sequence generator
// (per Applied Cryptography,
// by Bruce Schneier, Wiley, 1996)
```

- BallotResults.cpp
Diebold Election Systems

"Unfortunately, linear congruential generators cannot be used for cryptography"

- Page 369,
Applied Cryptography
by Bruce Schneier

Software Engineering

- Operating system bugs?
- Use third party plugin software
  - *Fmod* audio library
  - Not available for review
- They should use snprintf instead of printf, but do not.
- C++. Unsafe. In fact, buffer overflows gave (non-security) problems.
- Badly commented, no bug tracking etc.
- No audit process to prevent insertion of malicious code by programmer.

Comments dealing with Audio

"this is a bit of a hack for now."

AudioPlayer.cpp

"the BOOL beeped flag is a hack so we don't beep twice. This is really a result of the key handling being gorped."

WriteIn.cpp

"the way we deal with audio here is a gross hack."

BallotSelDlg.cpp

"need to work on exception *caused by audio*. I think they will currently result in double-fault."

BallotDlg.cpp

Code Fragment

```
void CBallotRelSet::Open(const CDistrict* district, const CBaseunit* baseunit,
const CVGroup* vgroup1, const CVGroup* vgroup2)
{
  ASSERT(m_pDB != NULL);
  ASSERT(m_pDB->IsOpen());
  ASSERT(GetSize() == 0);
  ASSERT(district != NULL);
  ASSERT(baseunit != NULL);
  if (district->KeyId() == -1) {
    Open(baseunit, vgroup1);
  } else {
    const CDistrictItem* pDistrictItem = m_pDB->Find(*district);
    if (pDistrictItem != NULL) {
      const CBaseunitKeyTable& baseunitTable = pDistrictItem->m_BaseunitKeyTable;
      int count = baseunitTable.GetSize();
      for (int i = 0; i < count; i++) {
        const CBaseunit& curBaseunit = baseunitTable.GetAt(i);
        if (baseunit->KeyId() == -1 || *baseunit == curBaseunit) {
          const CBallotRelationshipItem* pBalRelItem = NULL;
          while ((pBalRelItem = m_pDB->FindNextBalRel(curBaseunit, pBalRelItem))){
            if (!vgroup1 || vgroup1->KeyId() == -1 ||
               (*vgroup1 == pBalRelItem->m_VGroup1 && !vgroup2) ||
               (vgroup2 && *vgroup2 == pBalRelItem->m_VGroup2 &&
                *vgroup1 == pBalRelItem->m_VGroup1))
              Add(pBalRelItem);
          }
        }
      }
      m_CurIndex = 0;
      m_Open = TRUE;
    }
  }
}
```

Zero Comments

Diebold Redux

- Press release headline:
  - “Maryland Security Study Validates Diebold Election Systems Equipment for March Primary:Findings Consistent With Prior SAIC Review”
- Company President:
  - “Touch screen voting from Diebold Election Systems has evolved to be the most secure and accurate election system in the history of our democracy.”
- August 2007 Diebold Election Systems rebranded itself as “Premier Election Solutions”

Other Studies

- Science Applications International Corporation (SAIC) report
  - 2/3 of the report redacted
  - Executive summary:
    - “The system as implemented in policy, procedure, and technology, is at high risk of compromise.”
- Ohio report
  - cited “critical flaws” in top 4 vendors’ voting machines
- RABA report
  - ex-NSA red team consulting company
  - Executive Summary:
    - “The State of Maryland election system (comprising technical, operational, and procedural components), as configured at the time of this report, contains considerable security risks that can cause moderate to severe disruption in an election.”
- 2007 California Study:
  - http://www.sos.ca.gov/elections/elections_vsr.htm
  - Premier Election Solutions / Hart InterCivic / Sequoia Voting Systems
  - All of these are bad…

Recommendation #1

- Separate vote casting from tabulating
  - Touch screen machine produces paper ballot
    - need not be as trusted as today’s DREs
  - voter can use or destroy
  - scanning and tabulating machine
    - small code base
    - open source
    - extensive testing and certification
    - different manufacturer from touch screen

Recommendation #2

- Transparency
  - Require designs of machines to be public
  - Require security audit of machines by qualified experts
    - Require public report of this audit
  - Require open source for vote tabulation code
    - necessary but not sufficient

Recommendation #3

- Quality control
  - Establish criteria for testing the expertise of manufacturers
    - NIST could play this role
  - Require source code analysis for certification
  - Establish standards for policies and procedures
    - Aim for simplicity:
      - The more complicated and burdensome, the less likely to be followed