**Panel Session and Open Discussion**
**Join us for a wide-ranging debate on electronic voting, its risks, and its potential impact on democracy.**

# The E-voting Controversy: What are the Risks?

Wednesday April 19th • 7:00 pm – 9:00 pm

Maginnes Hall Room 102

Lehigh University, Bethlehem, PA

`http://www.cse.lehigh.edu/seminars/E-Voting.html`

*Sponsored in part by the Lehigh University Department of Computer Science and Engineering*

# Our participants

**Moderator**

- **Hannah Stewart-Gambino**
  Professor, Lehigh University Department of Political Science ▪ Director of Lehigh's Global Citizenship Program

**Panelists**

- **Christopher Borick**
  Associate Professor, Muhlenberg College Department of Political Science ▪ Director of Muhlenberg's Institute of Public Opinion
- **Bob Freeman**
  Pennsylvania State Representative ▪ Co-sponsor of H.B. 2000 to require a Voter Verified Paper Audit Trail (VVPAT)
- **Steve Freeman**
  Lecturer and Scholar, University of Pennsylvania Center for Organizational Dynamics ▪ Widely quoted researcher on polling discrepancies in contested elections, author of a forthcoming book on the 2004 election
- **Mary Ann Gould**
  Expert on managing change in corporate and private sectors ▪ Co-founder of the non-partisan Coalition for Voting Integrity
- **Daniel Lopresti**
  Associate Professor, Lehigh University Department of Computer Science and Engineering ▪ Noted computer security expert

# Setting the stage

- E-voting systems not as secure and transparent as they could be.
- Are they secure and transparent enough?

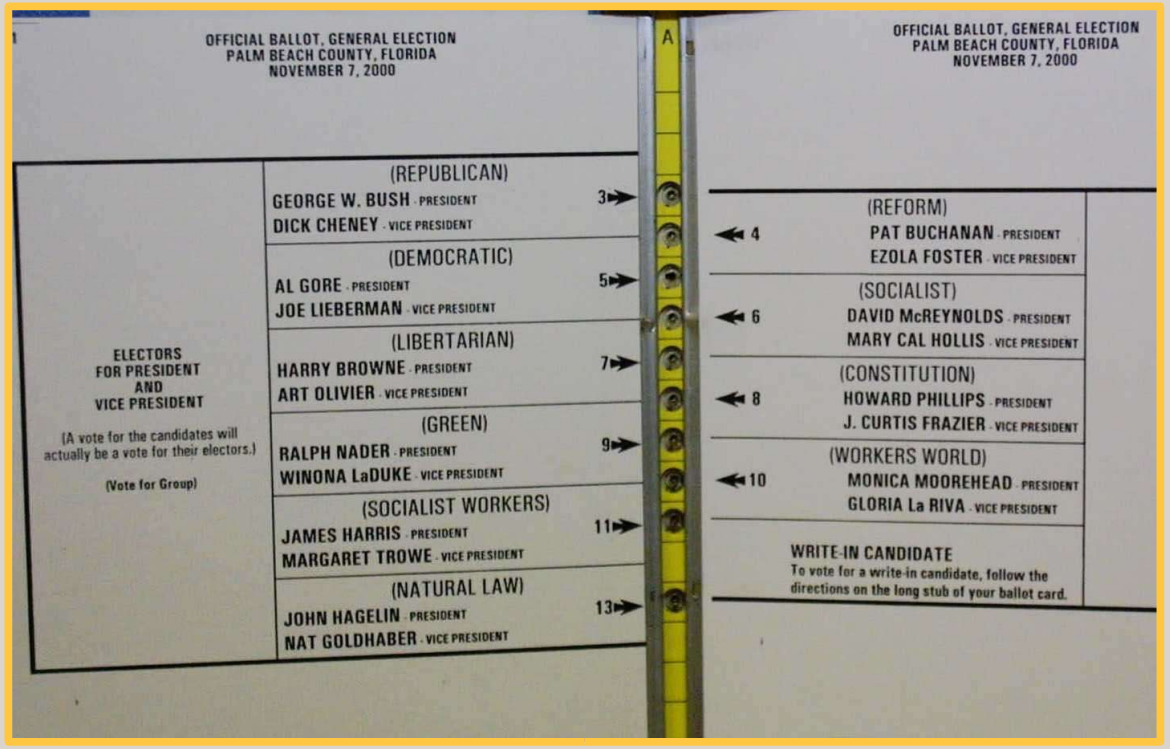
*This is something we must all decide for ourselves as citizens.*

- Any voting system carries with it some risk. Past experience with paper ballots, lever machines, etc., lets us understand that risk.
- What are the risks associated with e-voting technologies?

*This is the purpose of our panel session.*

# Background leading to HAVA

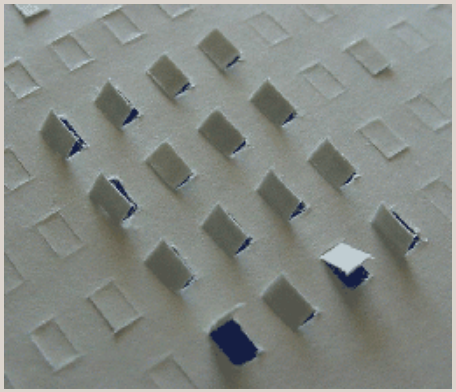
The infamous butterfly ballot from the 2000 Presidential election:

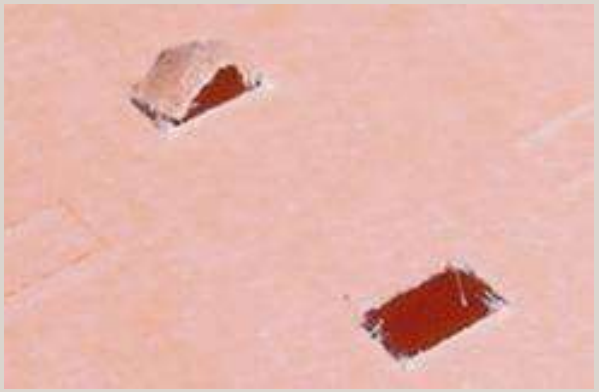



The Florida ballot is a classic example of bad user interface design. Computer software can suffer from such problems just as easily.

http://www2.indystar.com/library/factfiles/gov/politics/election2000/img/prezrace/butterfly_large.jpg

# Hanging chads & voter intent






Votomatic technology used in Florida was prone to paper jams. This led to hanging and dimpled chads, making it hard to determine voter intent.

http://www.cs.uiowa.edu/~jones/cards/chad.html
http://www.pushback.com/justice/votefraud/DimpledChadPictures.html

# Election technology & HAVA

The Help America Vote Act (HAVA) provides funds for states to replace punched card and lever voting systems. It does not mandate the use of direct recording electronic (DRE) systems.

Some general goals to keep in mind as we weigh alternatives:

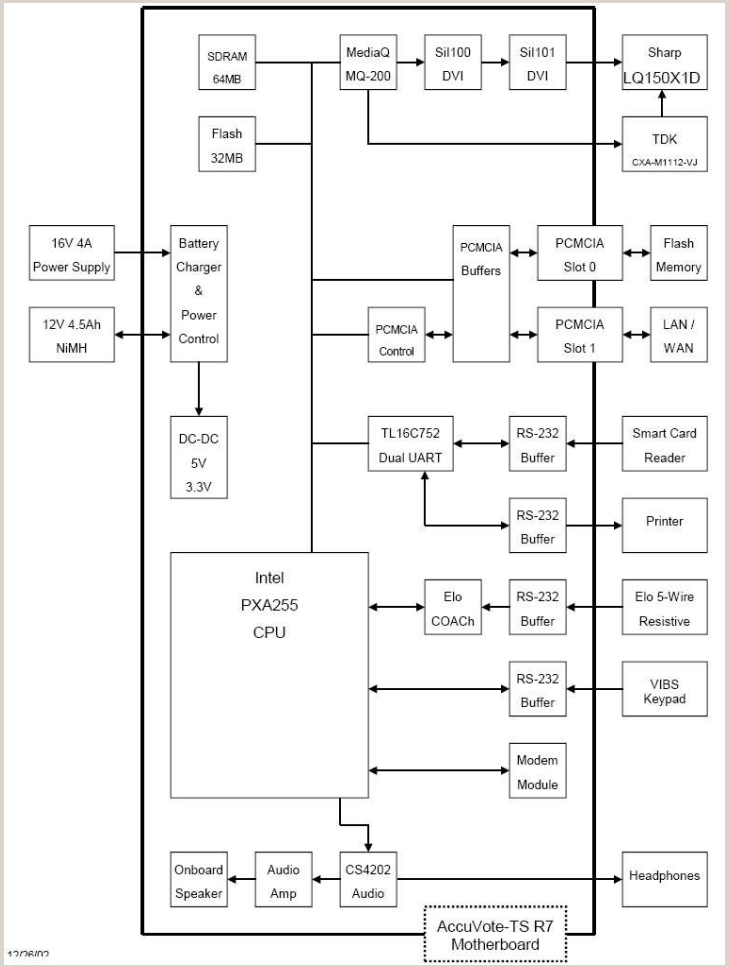
- secure and transparent elections,
- accurate determination of voter intent,
- voter anonymity,
- accessibility for disabled voters and non-native English voters,
- if possible, prevent overvoting (invalidates voter's ballot),
- if possible, prevent unintentional undervoting (voter confusion?).

http://www.fec.gov/hava/law_ext.txt

# Diebold AccuVote System

## Recent demo in Allentown:

## Diebold AccuVote-TSx block diagram:



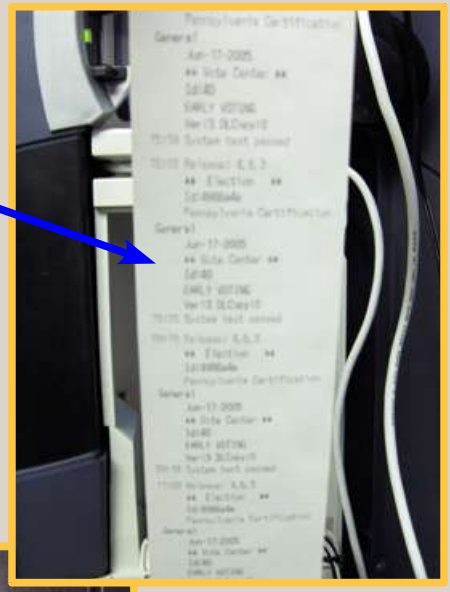

*DRE systems are nothing more than specialized computers.*

http://www.wfmz.com/cgi-bin/tt.cgi?action=viewstory&storyid=13711
http://www.bbvforums.org/forums/messages/1954/AccuVote-TSx_2_02_System_Overview-23267.pdf

# More photos from Diebold demo

*Built-in printer*

*Paper tape (used for end-of-day tally)*



*PCMCIA slot*

*PCMCIA card*

# E-voting risks

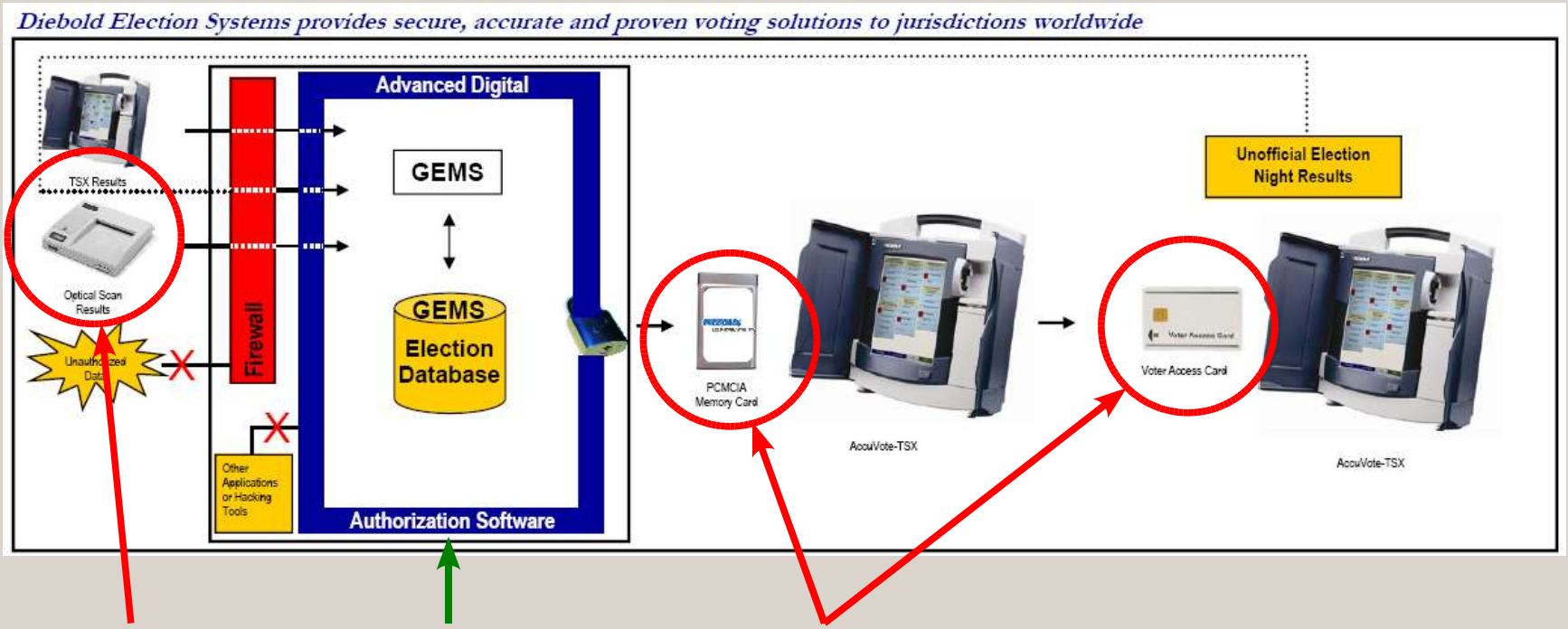
While there are several DRE vendors, one truth holds: all computer hardware/software systems of this complexity have bugs.

Bugs can manifest themselves in different ways:

- cause system to be unreliable (crash, lose votes),
- create openings that allow an outsider to compromise election,
- create openings that allow an inside to compromise election.

*Such attacks can be impossible to detect after-the-fact.*

# Diebold security




***What we mostly worry about***

***Probably pretty safe***

***What we mostly worry about***

(But insider attacks can arise anywhere.)

http://www.diebold.com/dieboldes/pdf/industrysecurity.pdf

# Risk analysis of e-voting software

- Avi Rubin and colleagues at Johns Hopkins obtained copy of Diebold e-voting software which appeared on the Internet.*
- Studied it carefully – made results public in 2003.
- Findings include:
  - "... far below even the most minimal security standards ..."
  - "... unauthorized privilege escalation, incorrect use of cryptography, vulnerabilities to network threats, ..."
  - "... voters ... can cast unlimited votes without being detected ..."

* E-voting vendors often assert they must be allowed to keep their software secret to protect it. This proves the futility of that idea.

"Analysis of an Electronic Voting System," Tadayoshi Kohno, Adam Stubblefield, Aviel D. Rubin, and Dan S. Wallach, *IEEE Symposium on Security and Privacy*, 2004.

# Risk analysis of e-voting software

## Summary of potential vulnerabilities identified by Rubin, et al.

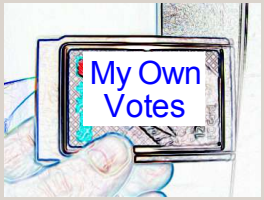
| | Voter (with forged smartcard) | Poll Worker (with access to storage media) | Poll Worker (with access to network traffic) | Internet Provider (with access to network traffic) | OS Developer | Voting Device Developer | Section |
|---|---|---|---|---|---|---|---|
| Vote multiple times using forged smartcard | • | • | • | | | | 3.2 |
| Access administrative functions or close polling station | • | • | | | • | • | 3.3 |
| Modify system configuration | | • | | | • | • | 4.1 |
| Modify ballot definition (e.g., party affiliation) | | • | • | • | • | • | 4.2 |
| Cause votes to be miscounted by tampering with configuration | | • | • | • | • | • | 4.2 |
| Impersonate legitimate voting machine to tallying authority | | • | • | • | • | • | 4.3 |
| Create, delete, and modify votes | | • | • | • | • | • | 4.3, 4.5 |
| Link voters with their votes | | • | • | • | • | • | 4.5 |
| Tamper with audit logs | | • | | | • | • | 4.6 |
| Delay the start of an election | | • | • | • | • | • | 4.7 |
| Insert backdoors into code | | | | | • | • | 5.3 |

"Analysis of an Electronic Voting System," Tadayoshi Kohno, Adam Stubblefield, Aviel D. Rubin, and Dan S. Wallach, *IEEE Symposium on Security and Privacy*, 2004.

# One potential exploit

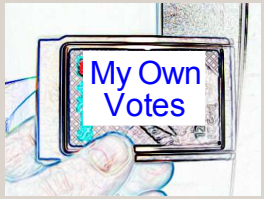

Attempt is made to protect integrity of voting records by encrypting them before storage on PCMCIA memory card ...

Okay!




No way!

... unfortunately, the key is hardwired in the code and now widely known across Internet (it's "F2654hD4").




Okay!

"Analysis of an Electronic Voting System," Tadayoshi Kohno, Adam Stubblefield, Aviel D. Rubin, and Dan S. Wallach, *IEEE Symposium on Security and Privacy*, 2004.

# A more recent risk analysis

- Report of the California Secretary of State's Voting Systems Technology Assessment Advisory Board (VSTAAB).
- Examined parts of both Diebold touchscreen system (AV-TX) and optical scan system (AV-OS) – published February 14, 2006.
- Findings include:
  - "Memory card attacks are a real threat ..."
  - "... anyone who has access to a memory card of the AV-OS ... and can have the modified card used ... can indeed modify the election results ..."
  - "The fact that the the [sic] results are incorrect cannot be detected except by a recount of the original paper ballots."

"Security Analysis of the Diebold AccuBasic Interpreter" by David Wagner, David Jefferson, Matt Bishop, Chris Karlof, and Naveen Sastry, February 14, 2006.

# A more recent risk analysis

| | Type | Impact |
|---|---|---|
| V1 | Array bounds violation | Overwrite any memory address within $\pm 2^{15}$ bytes of the global context structure with a 2-byte value that the adversary has partial control over. Might allow attacker to inject malicious code and take complete control of the machine. Might allow overwriting vote counters. |
| V2 | Format string vulnerability | Crash the machine; read the contents of memory within a narrow range |
| V3 | Input validation error | Choose any location on the memory card and begin executing it as .abo code; could be used to conceal malicious .abo code in unexpected locations, or to crash the machine. |
| V4 | Array bounds violation | Memory corruption; crash the machine. |
| V5 | Double-free() vulnerability | Overwrite any desired 4-byte memory address with any desired 4-byte value. Allows attacker to inject malicious code and take complete control of the machine. |
| V6 | Array bounds violation | Memory corruption: overwrite any memory address up to $2^{16}$ bytes after the global context structure with a 2-byte value that the adversary has no control over. Might allow overwriting vote counters. |
| V7 | Buffer overrun | Memory corruption; crash the machine |
| V8 | Buffer overrun, integer conversion bug | Memory corruption: overwrite up to $2^{15}$ consecutive bytes of memory starting at global context structure. Might allow attacker to inject malicious code and take complete control of the machine. Might allow overwriting vote counters. Information disclosure: read any memory location $\pm 2^{15}$ bytes away from global context structure. Crash the machine. |
| V9 | Buffer underrun | Memory corruption: overwrite up to $2^{15}$ consecutive bytes of memory extending backwards from the global context structure. Might allow attacker to inject malicious code and take complete control of the machine. Might allow overwriting vote counters. Information disclosure: read any memory location within this window. Crash the machine. |
| V10 | Buffer overrun | Overwrite return address on the stack. Allows attacker to inject malicious code and take complete control of the machine. |
| V11 | Array bounds violation | Information disclosure: read from potentially any memory address. Crash the machine. |
| V12 | Array bounds violation | Write any 2-byte value to any address up to $2^{16}$ bytes after the global context structure. Might allow attacker to inject malicious code and take complete control of the machine. Might allow overwriting vote counters. |
| V13 | Array bounds violation | Information disclosure: Read any 2-byte value from any address up to $2^{16}$ bytes after the global context structure. |
| V14 | Pointer arithmetic error | Crash machine. Could begin interpreting random memory locations as though they were .abo code. |
| V15 | Unchecked string operation | Machine might crash or become unresponsive |
| V16 | Unchecked string operation | Overwrite stack memory. Might allow attacker to inject malicious code and take complete control of the machine. |

*for AV-OS*

Summaries of potential vulnerabilities identified by Bishop, et al.

| | Type | Impact |
|---|---|---|
| W1 | Array bounds violation | Overwrite any memory address with a 4-byte value that the adversary has partial control over. Allows attacker to inject malicious code and take complete control of the machine. |
| W3 | Input validation error | Choose any memory location and begin executing it as .abo code; could be used to conceal malicious .abo code in unexpected locations, or to crash the machine. |
| W6 | Array bounds violation | Overwrite any memory location with any desired value. Allows attacker to inject malicious code and take complete control of the machine. |
| W7 | Buffer overrun | Memory corruption; crash the machine |
| W8 | Buffer overrun, integer conversion bug | Corrupts memory until the machine crashes. |
| W10 | Buffer overrun | Overwrite return address on the stack. Allows attacker to inject malicious code and take complete control of the machine. |
| W11 | Array bounds violation | Information disclosure: read from potentially any memory address. Crash the machine. |
| W12 | Array bounds violation | Writes any 4-byte value to any address. Allows attacker to inject malicious code and take complete control of the machine. |
| W13 | Array bounds violation | Information disclosure: read a 4-byte value from any address. |
| W14 | Pointer arithmetic error | Crash machine. Could begin interpreting random memory locations as though they were .abo code. |

*for AV-TX*

"Security Analysis of the Diebold AccuBasic Interpreter" by David Wagner, David Jefferson, Matt Bishop, Chris Karlof, and Naveen Sastry, February 14, 2006.

# Some lessons never learned

"There is a serious flaw in the key management of the crypto code that otherwise should protect the AV-TSx from memory card attacks. Unless election officials avail themselves of the option to create new cryptographic keys, the AV-TSx uses a default key. This key is hard coded into the source code for the AV-TSx, which is poor security practice because, among other things, it means the same key is used in every such machine in the U.S. Worse, the particular default key in question was openly published two and a half years ago in a famous research paper, and is now known by anyone who follows election security, and can be found through Google."

"Security Analysis of the Diebold AccuBasic Interpreter" by David Wagner, David Jefferson, Matt Bishop, Chris Karlof, and Naveen Sastry, February 14, 2006.

# Common retorts

- "These attack scenarios are unlikely."
- "Our e-voting systems are certified, so they must be safe."
- "Poll workers are trained to recognize potential problems."
- "Multiple copies of the data are stored in the system, so we're okay."
- "Re-printing the end-of-day tally is just as good as a recount."
- "There's no evidence of anyone having success in an attack like this."

My assessment: = optimistic = wrong = plain silly

*There is no doubt we need good policies and procedures in addition to good, safe technology. (I believe almost everyone involved would like to do the right thing.)*

# My recommendations

For secure and transparent elections, we should insist on:

- Giving independent experts unfettered access to e-voting software and hardware for verification purposes.
- A Voter Verified Paper Audit Trail (VVPAT).

And tell our lawmakers to pass pending legislation:

- H.R. 550 ("The Voter Confidence and Increased Accessibility Act")
- Pennsylvania H.B. 2000