# Week 12: Lecture A

## Election Cybersecurity

Tuesday, November 11, 2025

# Announcements

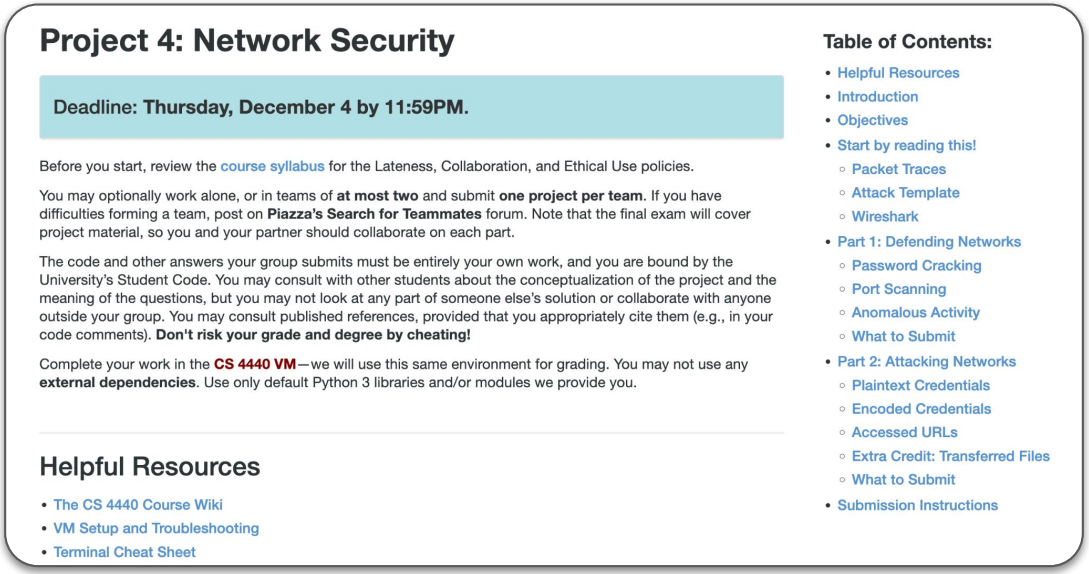
- **Project 3** grades are now available on **Canvas**

- Think we made an error? Request a regrade!
  - Valid regrade requests:
    - You have verified your solution is correct (i.e., we made an error in grading)

**Project 3 Regrade Requests** (see **Piazza** pinned link):
Submit by **11:59 PM** on **Monday 11/17** via **Google Form**

# Announcements

- **Project 4: NetSec** released
  - **Deadline:** Thursday, December 4th by 11:59PM



## Project 4: Network Security

**Deadline: Thursday, December 4 by 11:59PM.**

Before you start, review the course syllabus for the Lateness, Collaboration, and Ethical Use policies.

You may optionally work alone, or in teams of **at most two** and submit **one project per team**. If you have difficulties forming a team, post on **Piazza's Search for Teammates** forum. Note that the final exam will cover project material, so you and your partner should collaborate on each part.

The code and other answers your group submits must be entirely your own work, and you are bound by the University's Student Code. You may consult with other students about the conceptualization of the project and the meaning of the questions, but you may not look at any part of someone else's solution or collaborate with anyone outside your group. You may consult published references, provided that you appropriately cite them (e.g., in your code comments). **Don't risk your grade and degree by cheating!**

Complete your work in the **CS 4440 VM**—we will use this same environment for grading. You may not use any **external dependencies**. Use only default Python 3 libraries and/or modules we provide you.

### Helpful Resources

- The CS 4440 Course Wiki
- VM Setup and Troubleshooting
- Terminal Cheat Sheet

**Table of Contents:**

- Helpful Resources
- Introduction
- Objectives
- Start by reading this!
  - Packet Traces
  - Attack Template
  - Wireshark
- Part 1: Defending Networks
  - Password Cracking
  - Port Scanning
  - Anomalous Activity
  - What to Submit
- Part 2: Attacking Networks
  - Plaintext Credentials
  - Encoded Credentials
  - Accessed URLs
  - Extra Credit: Transferred Files
  - What to Submit
- Submission Instructions

# Project 4 Progress

Working on Part 1

0%

Finished Part 1, working on Part 2

0%

Finished both Part 1 and Part 2

0%

None of the above

0%

# Announcements

utahsec

See Discord for meeting info!

**utahsec.cs.utah.edu**

Kahlert School of Computing

# Graduate Program Open House

# Why PhD?

**Information session for prospective graduate students**

- What to expect from graduate school
- Reasons to pursue graduate career
- Perspective of alumni and current students
- How to prepare your application (and a statement of purpose)

**RSVP / Zoom links:**

## November 14, 3:00pm – 5:00pm

**MEB 3147 (LCR) and Zoom ( free pizza—please RSVP )**

# Announcements

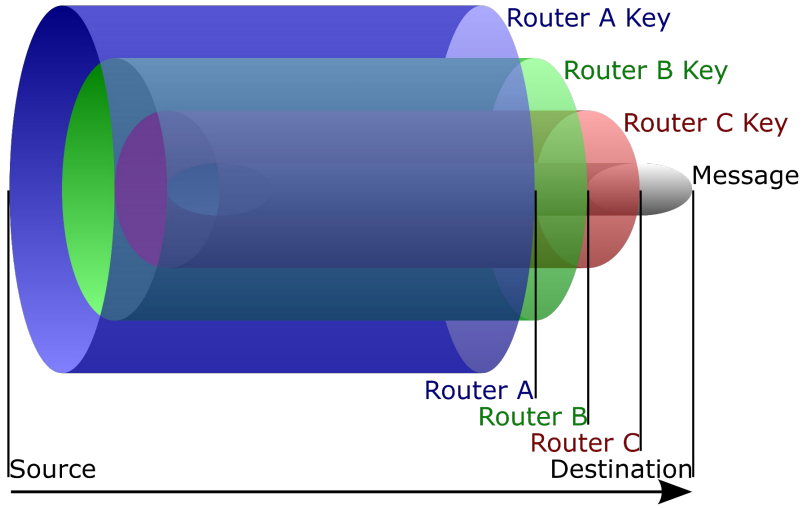
- Instructor on work travel next week
  - Presenting our **GUI fuzzing work** at ASE'25 😀
- Guest lectures planned for both days
  - **Week 13A:** Cyber-physical Systems Security
    - **Guest speaker:** Dr. Luis Garcia (Asst. Prof @ UofU)
  - **Week 13B:** Binary Reverse Engineering
    - **Guest speaker:** Zao Yang (researcher in my group)
- Attendance **not graded** for these lectures…
  - But you should definitely show up
  - **These are major hot topics in security!**

# Last time on CS 4440...

Security in Practice:
Tor—The Onion Router

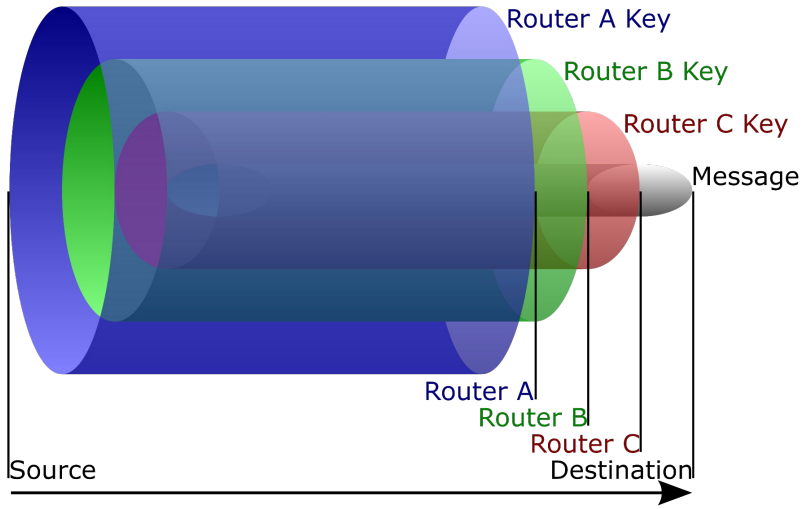
# Anonymity Primitive: Onion Routing

- Each message is ???



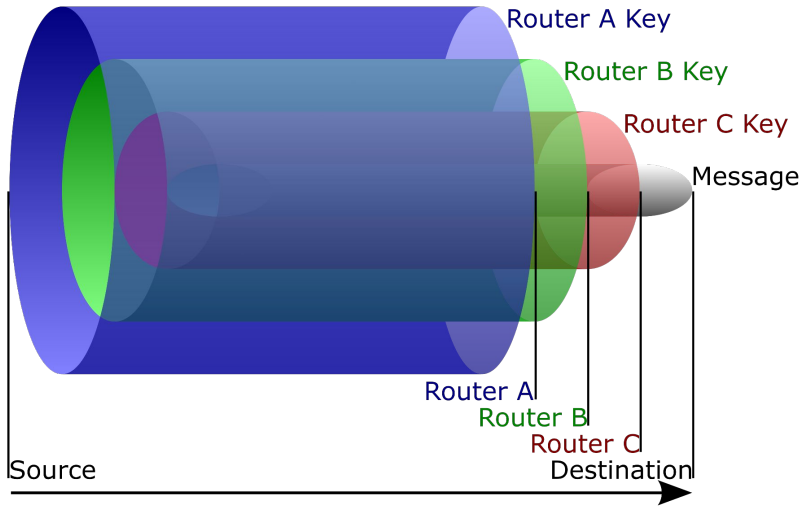
# Anonymity Primitive: Onion Routing

- Each message is **repeatedly encrypted**
  - **Analogy:** multiple layers of an onion
- Sent through **multiple network nodes**
  - These nodes are called **onion routers**
  - Each node removes an encryption layer to uncover the message **routing instructions**
  - Process repeats when sent to next router
- **Anonymity:** prevents **???**

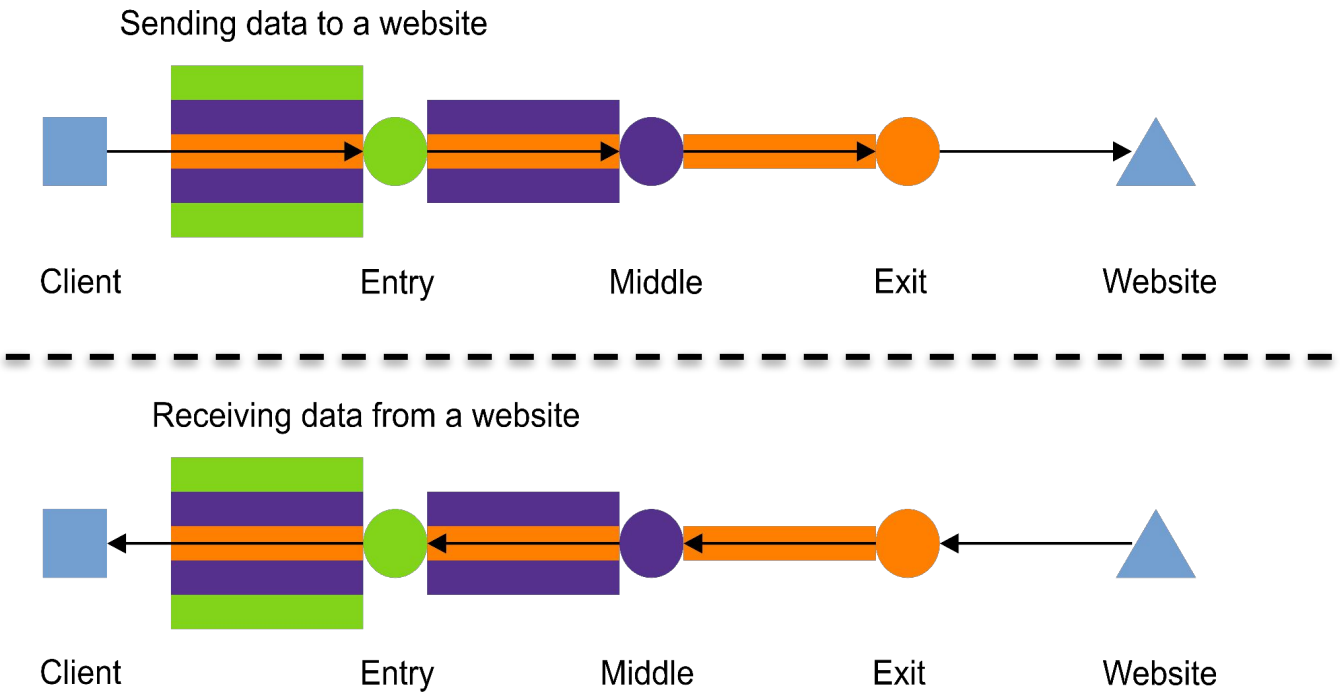


# Anonymity Primitive: Onion Routing

- Each message is **repeatedly encrypted**
  - **Analogy:** multiple layers of an onion
- Sent through **multiple network nodes**
  - These nodes are called **onion routers**
  - Each node removes an encryption layer to uncover the message **routing instructions**
  - Process repeats when sent to next router
- **Anonymity:** prevents any intermediary nodes from knowing message **origin**, **destination**, and **contents**

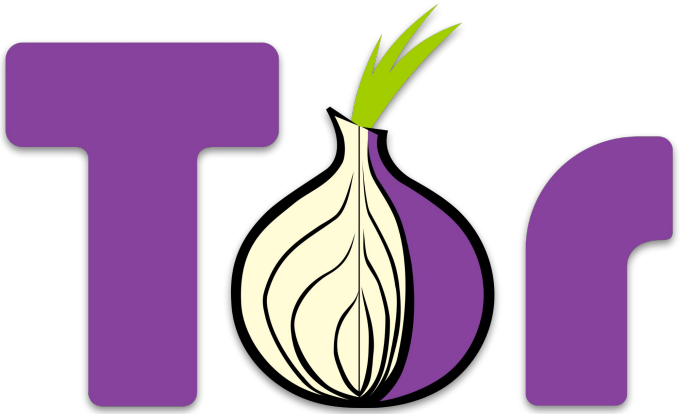


# Onion Routing Visualized



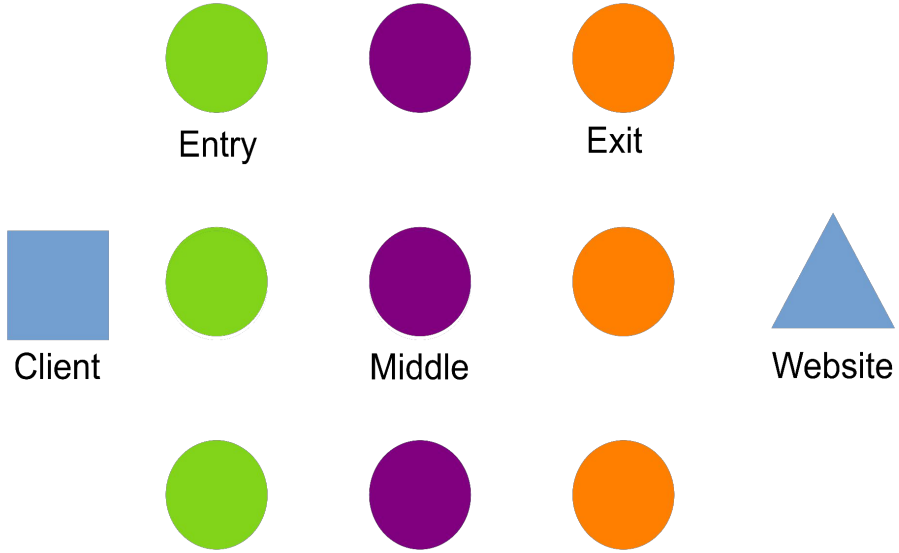
# Tor: The Onion Router

- **Tor:** a distributed overlay network
  - Anonymizes TCP-based applications
    - Secure shell
    - Web browsing
    - Instant messaging



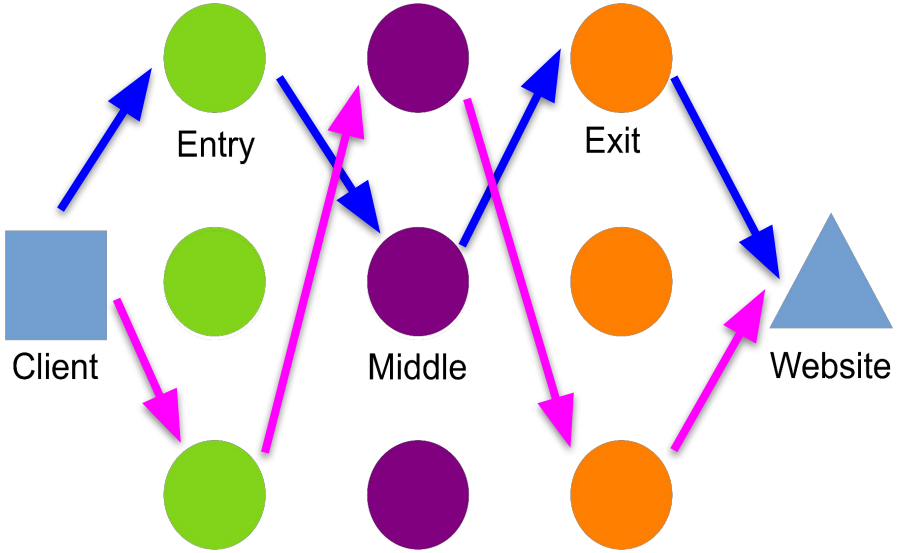
# Tor: The Onion Router

- **Tor:** a distributed overlay network
  - Anonymizes TCP-based applications
    - Secure shell
    - Web browsing
    - Instant messaging
- Clients choose ???



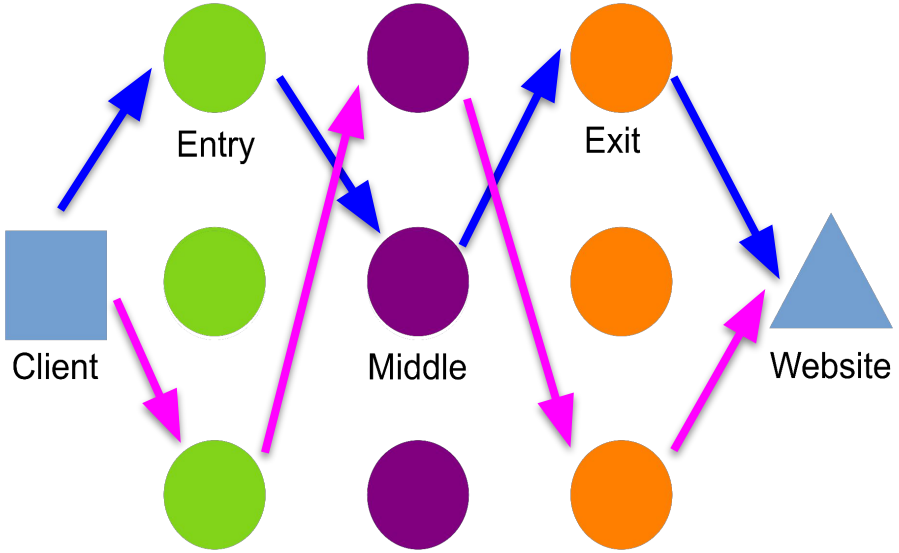
# Tor: The Onion Router

- **Tor:** a distributed overlay network
  - Anonymizes TCP-based applications
    - Secure shell
    - Web browsing
    - Instant messaging
- Clients choose the **circuit paths**
  - Messages unwrapped at each onion router using a symmetric key
- Onion routers only know **???**

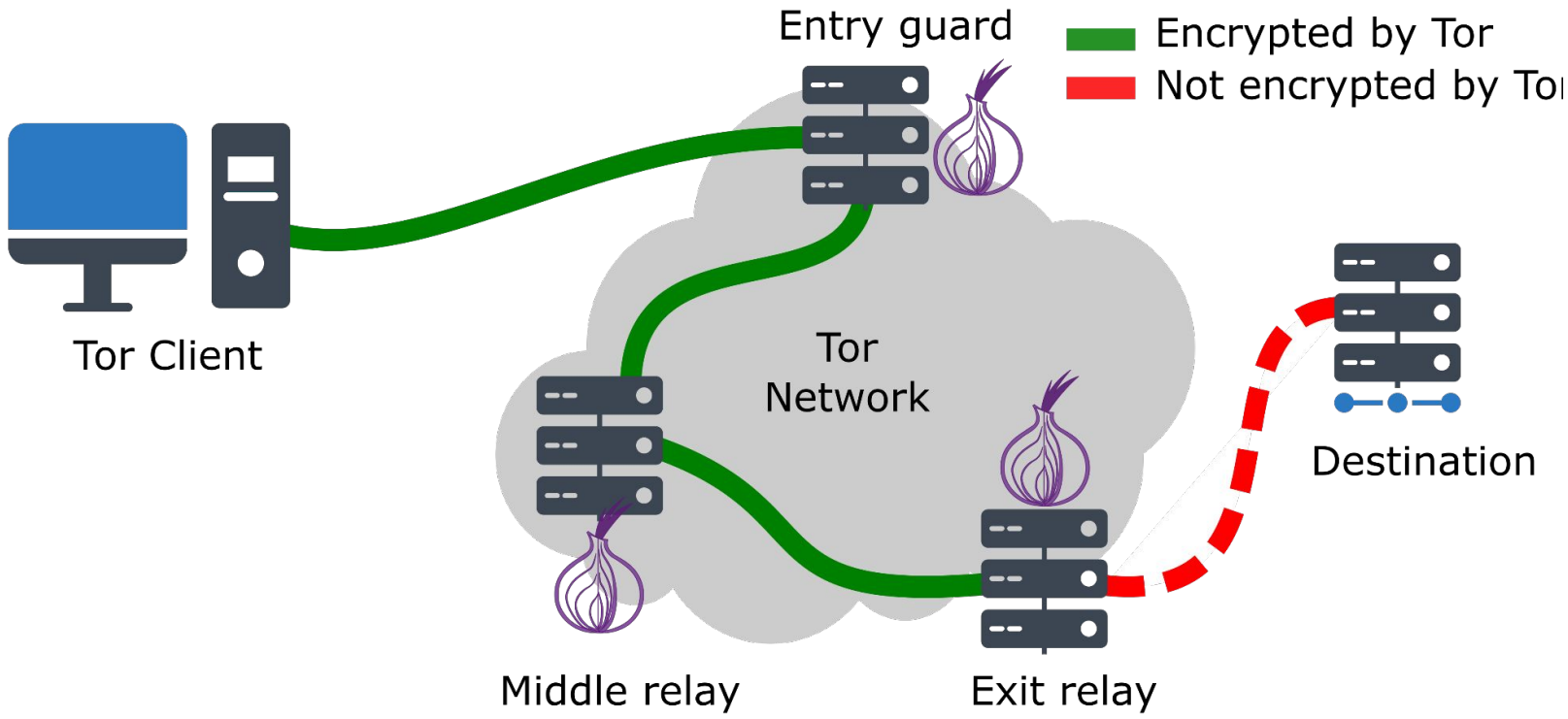


# Tor: The Onion Router

- **Tor:** a distributed overlay network
  - Anonymizes TCP-based applications
    - Secure shell
    - Web browsing
    - Instant messaging
- Clients choose the **circuit paths**
  - Messages unwrapped at each onion router using a symmetric key
- Onion routers only know their **successor** or **predecessor** nodes
  - They don't know of any other nodes



# How Tor Works



# Attacking Tor

- **Possible attacks against Tor?**

# Attacking Tor

- **Possible attacks against Tor?**
- **Leak DNS requests** when they aren't transmitted via Tor
- Perform **volume/timing analysis** to characterize behavior
- **Add malicious nodes** to intercept unencrypted exit traffic

# Attacking Tor

- **Possible attacks against Tor?**
- **Leak DNS requests** when they aren't transmitted via Tor
  - **Defense: ???**
- Perform **volume/timing analysis** to characterize behavior
  - **Defense: ???**
- **Add malicious nodes** to intercept unencrypted exit traffic
  - **Defense: ???**

# Attacking Tor

- **Possible attacks against Tor?**

- **Leak DNS requests** when they aren't transmitted via Tor
  - **Defense:** enforce all DNS requests **through Tor encryption**

- Perform **volume/timing analysis** to characterize behavior
  - **Defense:** inject **noisy data** to throw off analysis heuristics

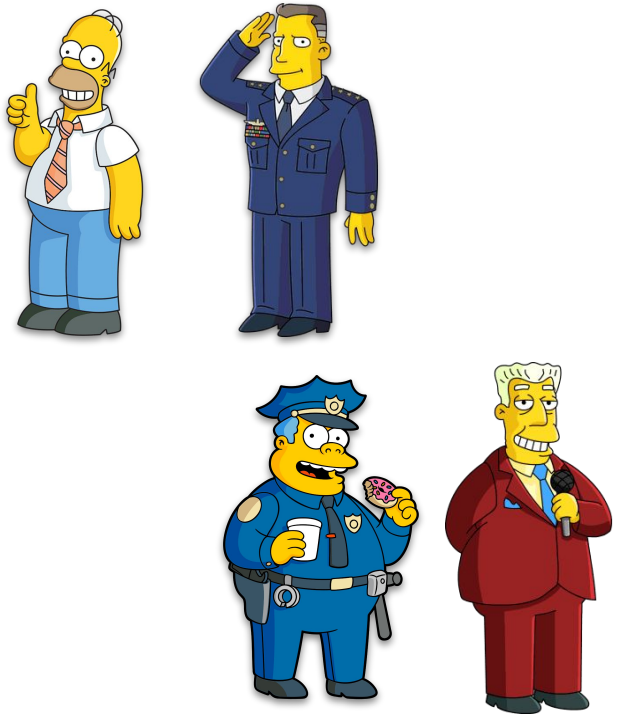
- **Add malicious nodes** to intercept unencrypted exit traffic
  - **Defense:** never use unencrypted protocols—**use HTTPS**

# Who uses Tor?

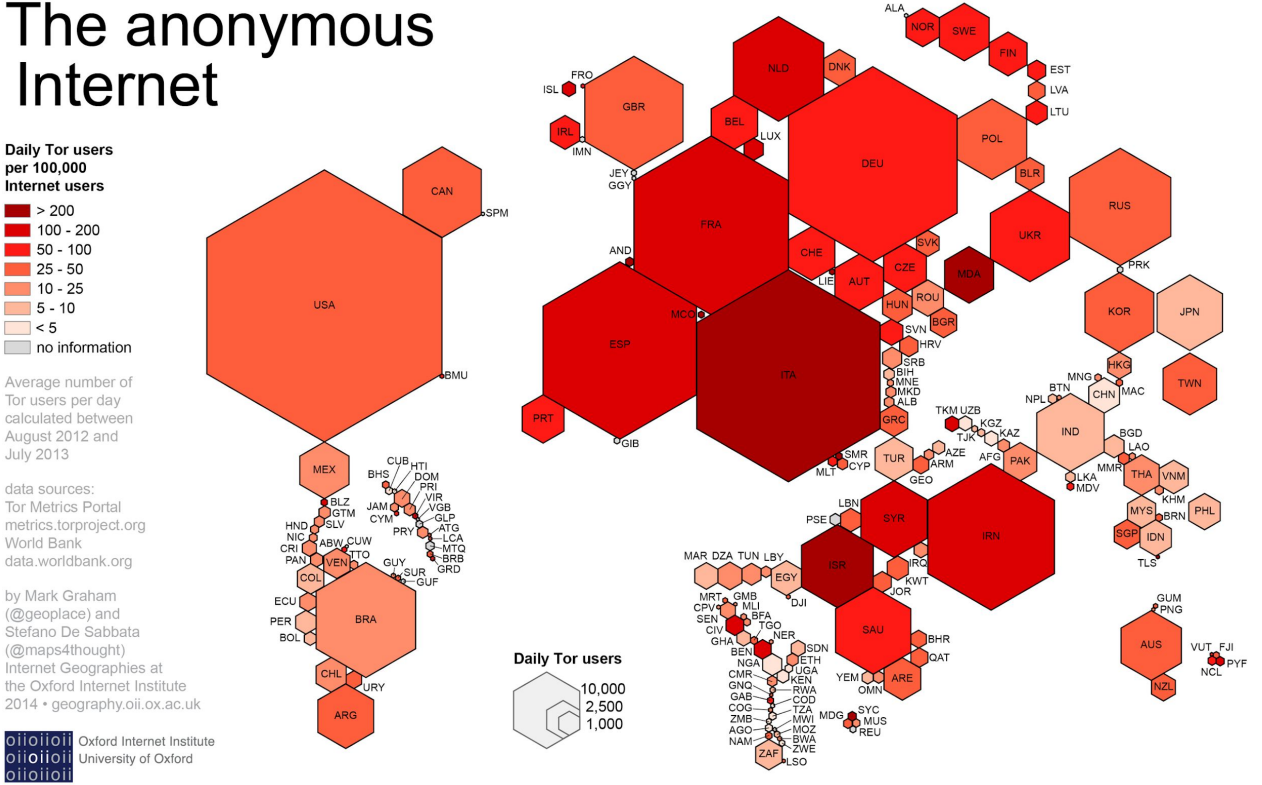
- ???

# Who uses Tor?

- **Normal People**
  - Privacy-conscious folks
- **Intelligence Agencies**
  - Secret agents in the field
- **Law Enforcement**
  - Online “undercover” operations
- **Journalists and Bloggers**
  - Citizen journalists inspiring social change
- **Activists and Whistleblowers**
  - Raising their voice and avoiding persecution
- **White-hat and Black-hat Hackers**
  - And everyone in between!



# Who uses Tor?


The anonymous Internet
Daily Tor users per 100,000 Internet users
> 200
100 - 200
50 - 100
25 - 50
10 - 25
5 - 10
< 5
no information
Average number of Tor users per day calculated between August 2012 and July 2013
data sources:
Tor Metrics Portal
metrics.torproject.org
World Bank
data.worldbank.org
by Mark Graham (@geoplace) and Stefano De Sabbata (@maps4thought)
Internet Geographies at the Oxford Internet Institute
2014 • geography.oii.ox.ac.uk
Oxford Internet Institute
University of Oxford
Daily Tor users
10,000
2,500
1,000

# What services get hidden?

# Positive Tor Use Cases

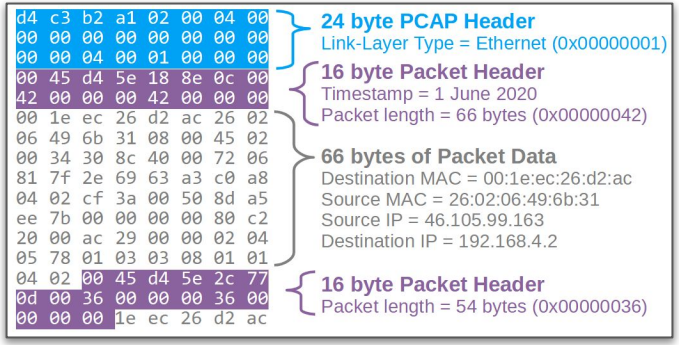
# Questions?

# Recap: Project 4 Overview

- Focuses on **network packet analysis**
  - Leveraging data contained within packets to achieve network defenses and attacks
- **Scenario:** helping a fictional university secure its enterprise campus network
  - Detect and characterizing likely attacks
  - Demonstrate how info can be intercepted

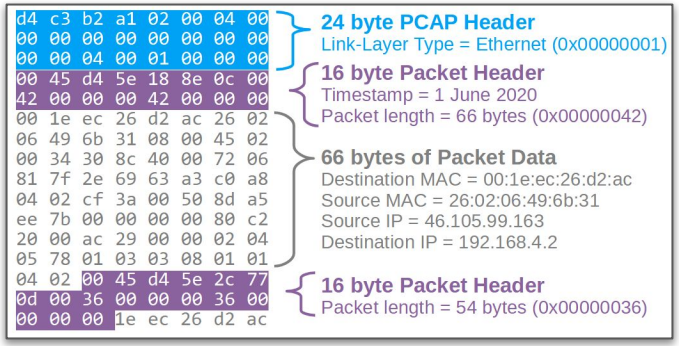
# Recap: Project 4 Overview

- We provide a series of network packet traces (**pcaps**)
  - **Your job:** write scripts to analyze them!
- **Part 1:** detecting **network attacks**
  - Password cracking, port scanning, SYN floods
- **Part 2:** stealing **sensitive information**
  - Unencrypted credentials, browsing history
  - **Extra credit:** stealing transfered files



# Recap: Project 4 Overview

- We provide a series of network packet traces (**pcaps**)
  - **Your job:** write scripts to analyze them!
- **Part 1:** detecting **network attacks**
  - Password cracking, port scanning, SYN floods
- **Part 2:** stealing **sensitive information**
  - Unencrypted credentials, browsing history
  - **Extra credit:** stealing transfered files
- You will use Python 3's **Scapy** library
  - A huge and powerful packet analysis API…
  - But we'll really only use **a few parts** of it



# Recap: Scapy Fundamentals

- Python API for programmatic packet capture and analysis
  - Think of it as **"Wireshark in API form"**
- We provide **skeleton code** template
  - Sets-up the packet parsing workflow

```
#!/usr/bin/python3
import logging
logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
from scapy.all import *
import re

def parsePacket(packet):
    if not packet.haslayer("TCP"): return
    # ----------------------------------------
    # TODO: finish implementing parsePacket()!
    # ----------------------------------------
    return

if __name__ == "__main__":
    for packet in rdpcap(sys.argv[1]):
        parsePacket(packet)
```

# Recap: Scapy Fundamentals

- Python API for programmatic packet capture and analysis
  - Think of it as **"Wireshark in API form"**
- We provide **skeleton code** template
  - Sets-up the packet parsing workflow
  - **Your job:** finish implementing the function `parsePacket()`

```
#!/usr/bin/python3
import logging
logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
from scapy.all import *
import re

def parsePacket(packet):
    if not packet.haslayer("TCP"): return
    # ----------------------------------------
    # TODO: finish implementing parsePacket()!
    # ----------------------------------------
    return

if __name__ == "__main__":
    for packet in rdpcap(sys.argv[1]):
        parsePacket(packet)
```

# Recap: Scapy Fundamentals

- Python API for programmatic packet capture and analysis
  - Think of it as **"Wireshark in API form"**
- We provide **skeleton code** template
  - Sets-up the packet parsing workflow
  - **Your job:** finish implementing the function `parsePacket()`
- You may also add **additional code**
  - E.g., global variables or data structures
  - E.g., printing functionality in `main()`

```
#!/usr/bin/python3
import logging
logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
from scapy.all import *
import re

def parsePacket(packet):
    if not packet.haslayer("TCP"): return
    # -------------------------------------------
    # TODO: finish implementing parsePacket()!
    # -------------------------------------------
    return

if __name__ == "__main__":
    for packet in rdpcap(sys.argv[1]):
        parsePacket(packet)
```

# Recap: Scapy Fundamentals

- Only a few things you'll need…
  - Get a packet's **TCP flags**:

```
packet["TCP"].flags
```

  - Get a packet's **destination port**

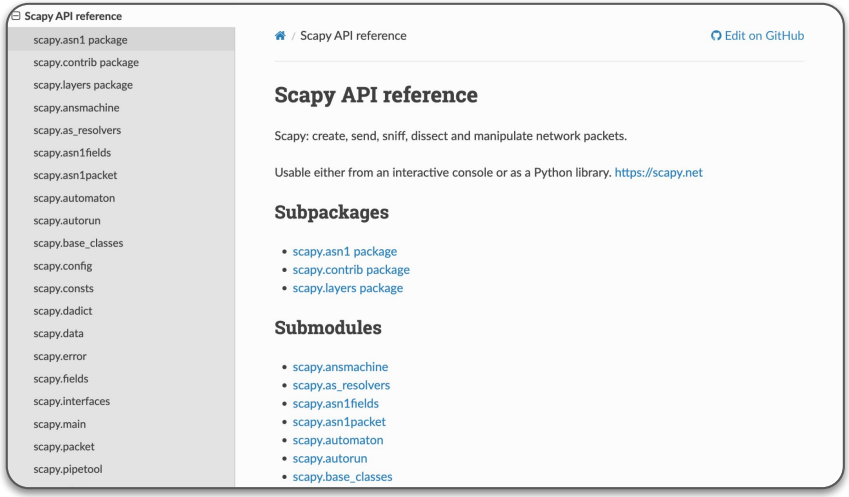
```
packet["TCP"].dport
```

  - Get a packet's **source IP address**

```
packet["IP"].src
```

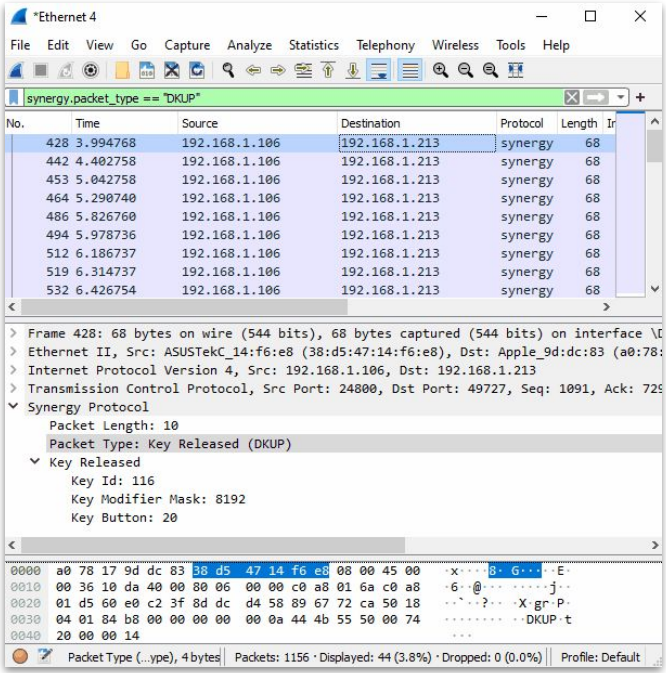
  - Get a packet's TCP **payload**:

```
bytes(packet["TCP"].payload).decode('utf-8','replace')
```



# Recap: Suggested Workflow

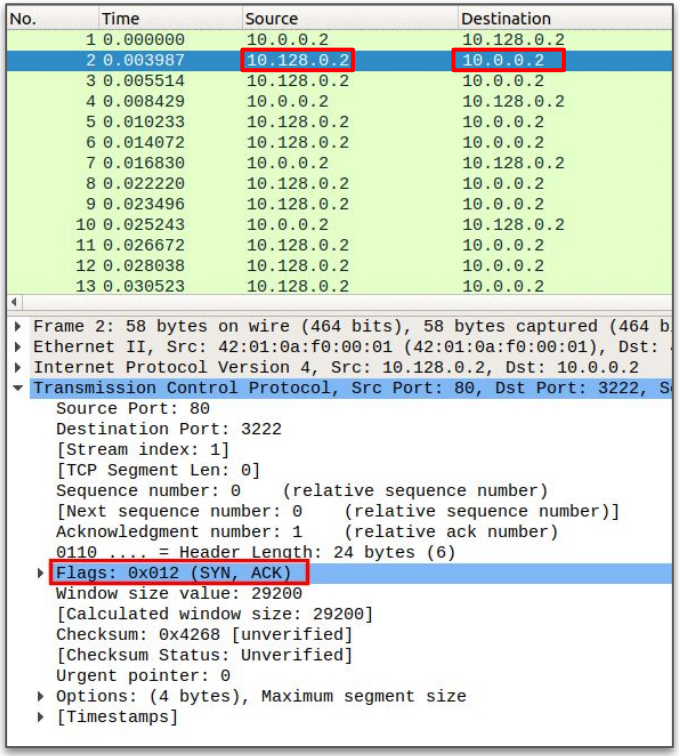
- Before you start writing a **Scapy** script, inspect the trace *manually* via **Wireshark**
  - Super helpful for viewing a packet's contents
  - Use this to bootstrap your script's approach!



# Recap: Suggested Workflow

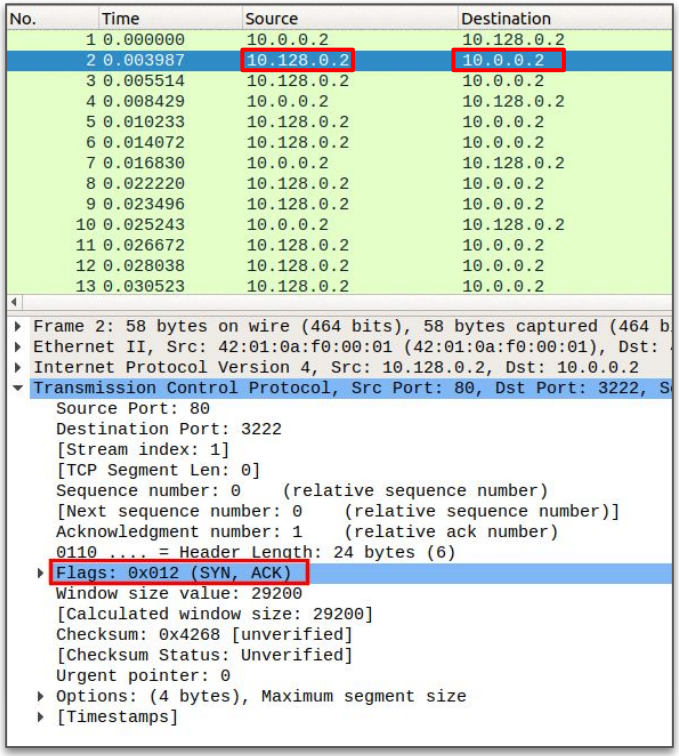
- Before you start writing a **Scapy** script, inspect the trace *manually* via **Wireshark**
  - Super helpful for viewing a packet's contents
  - Use this to bootstrap your script's approach!

- For each target, answer the following:
  - What **packet fields** matter?
  - How to **extract** relevant data?
  - How to **store and process** this data?



# Recap: Suggested Workflow

- Before you start writing a **Scapy** script, inspect the trace *manually* via **Wireshark**
  - Super helpful for viewing a packet's contents
  - Use this to bootstrap your script's approach!
- For each target, answer the following:
  - What **packet fields** matter?
  - How to **extract** relevant data?
  - How to **store and process** this data?
- Finalize your **high-level game plan** first!
  - Then start developing your solution scripts!



# Questions?

# This time on CS 4440...

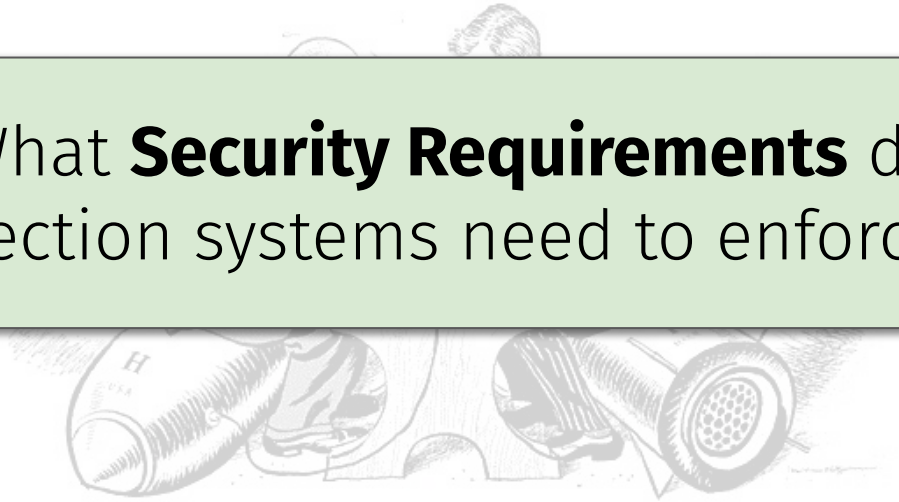
Election Cybersecurity
Voting Technology
Computerized Voting
Attacking Voting Systems

# Elections

- Why have them?

# Elections

- Why have them?



What **Security Requirements** do election systems need to enforce?

# What security requirements must election systems enforce?

Nobody has responded yet.

Hang tight! Responses are coming in.

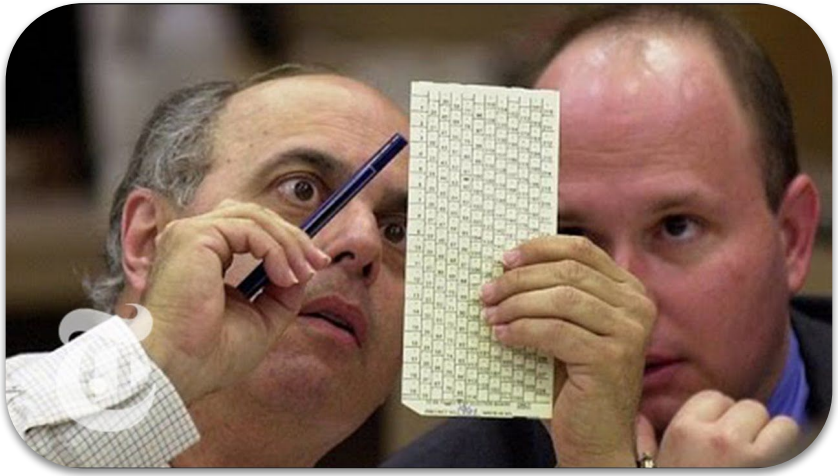
# Requirement #1: Integrity

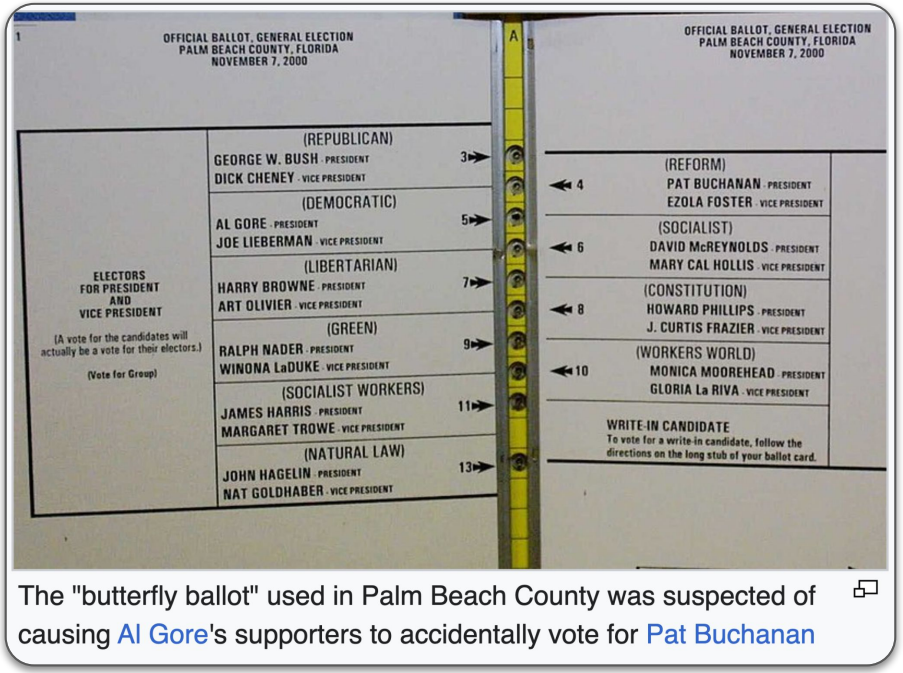
- **Goals: ???**

# Requirement #1: Integrity

- **Goals:** outcome matches voter's **intent**
  - Votes are cast as intended
  - Votes are counted as cast

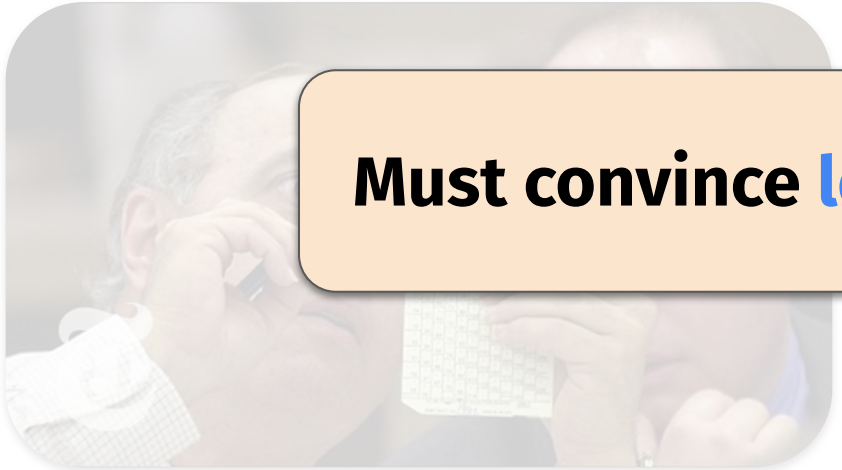
# Requirement #1: Integrity

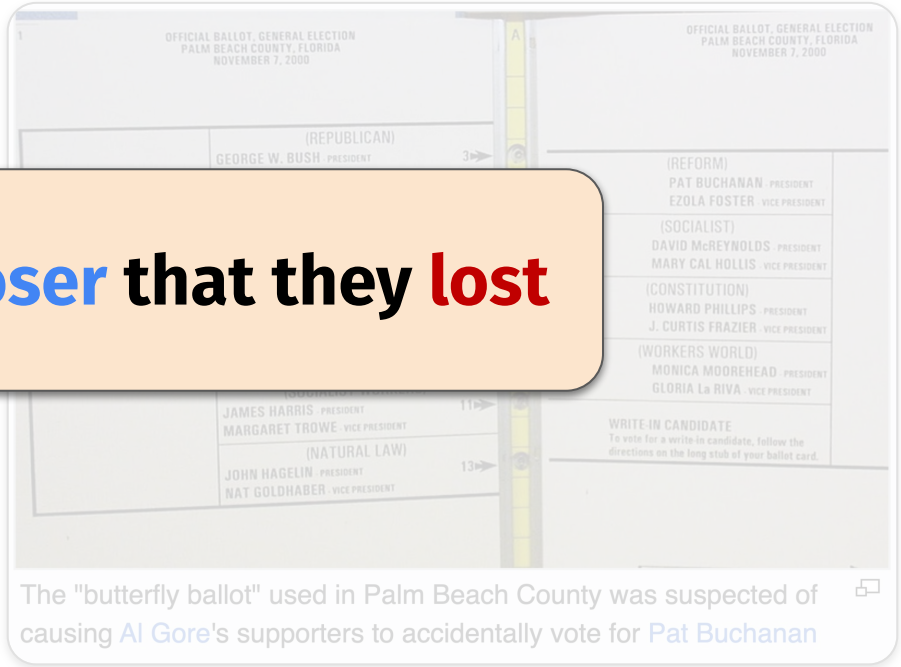





The "butterfly ballot" used in Palm Beach County was suspected of causing Al Gore's supporters to accidentally vote for Pat Buchanan

# Requirement #1: Integrity






The "butterfly ballot" used in Palm Beach County was suspected of causing Al Gore's supporters to accidentally vote for Pat Buchanan

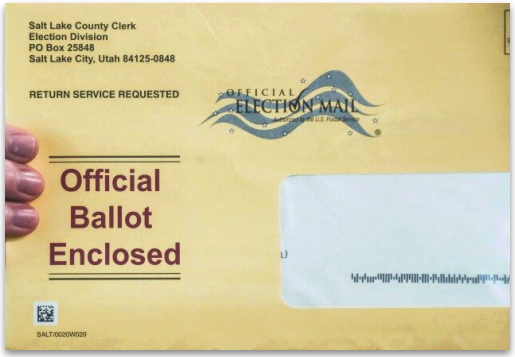
**Must convince loser that they lost**

# Requirement #2: Confidentiality

- **Goals: ???**

# Requirement #2: Confidentiality

- **Goals:** nobody can figure out **how** you voted
  - … even if you try to prove it to them

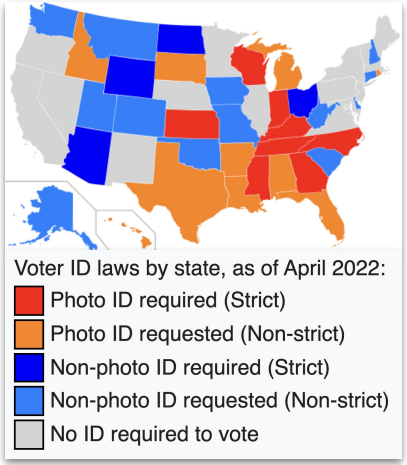




# Requirement #3: Authentication

- **Goals: ???**

# Requirement #3: Authentication

- **Goals:**
  - Only **authorized voters** can cast votes
  - Each voter can cast **at most one** vote



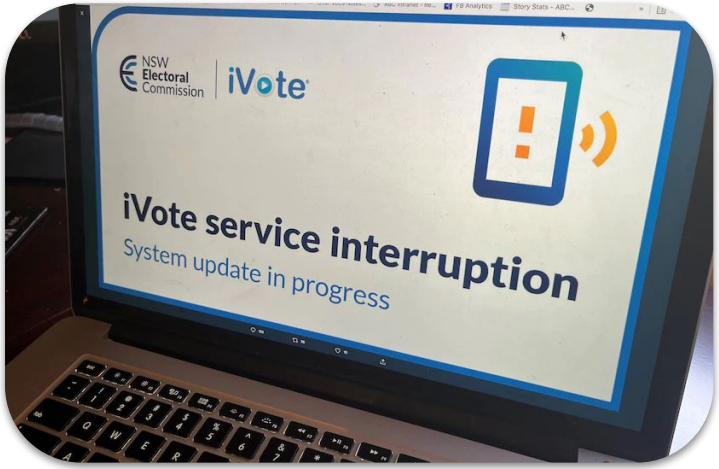


# Requirement #4: Availability

- **Goals: ???**

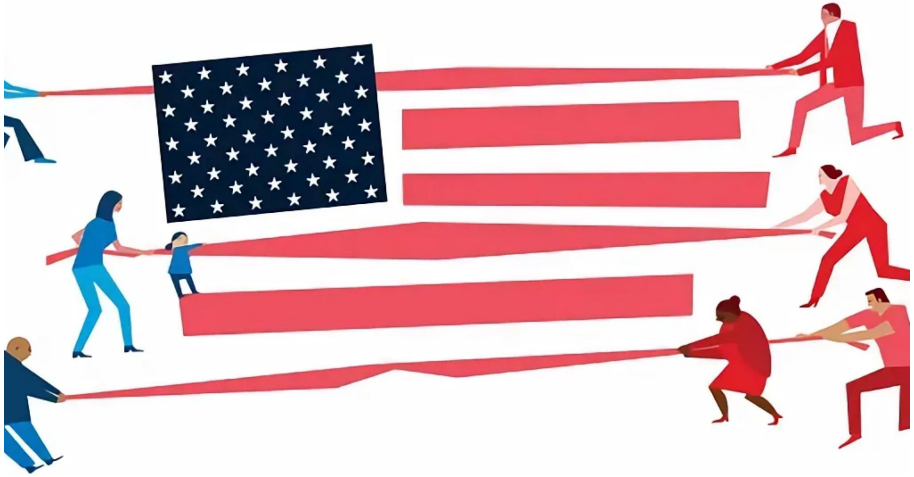
# Requirement #4: Availability

- **Goals:**
  - All authorized voters have **opportunity** to vote
  - System is able to **accept all votes** on schedule
  - System can produce results in a **timely manner**

# Tension Between these Properties

**Ballot Integrity** ◄- - - - - ► **Ballot Confidentiality**

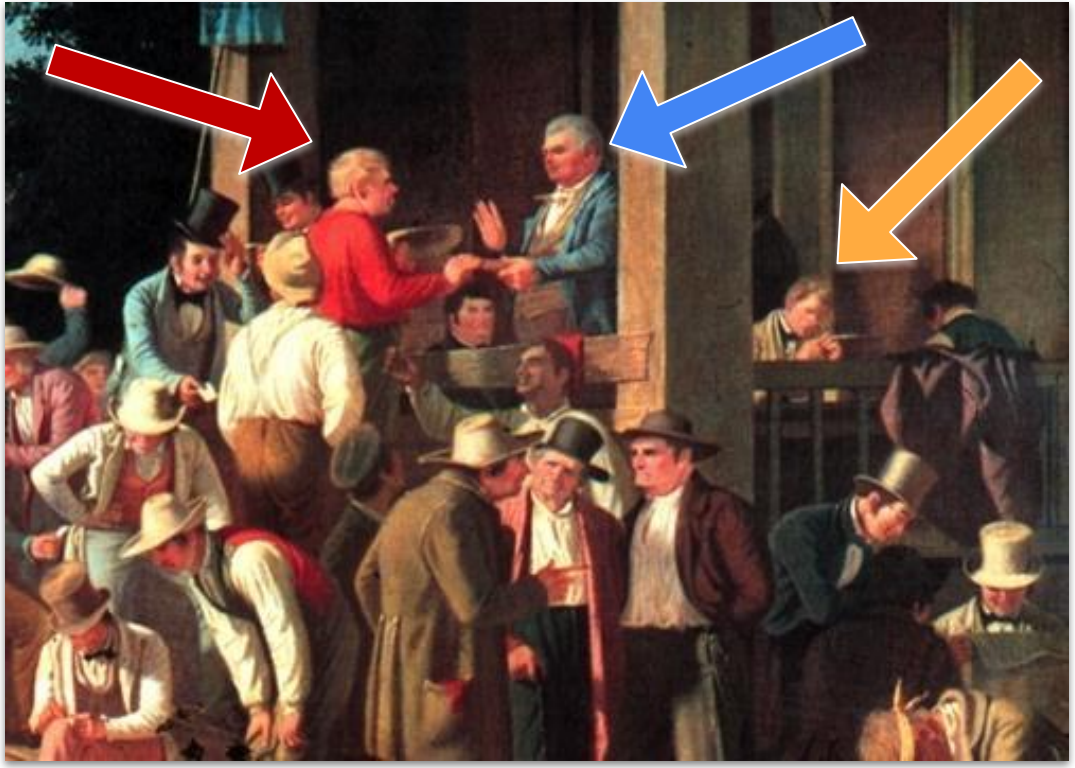


**Voting Availability** ◄- - - - - ► **Voter Authentication**

# Early Voting Technology

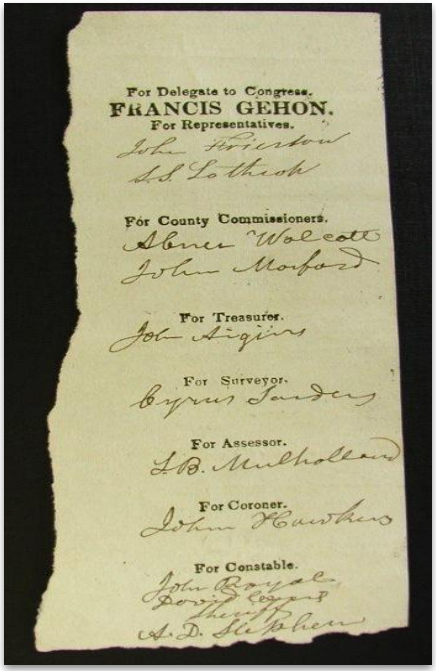
# Voice Voting

# Voice Voting

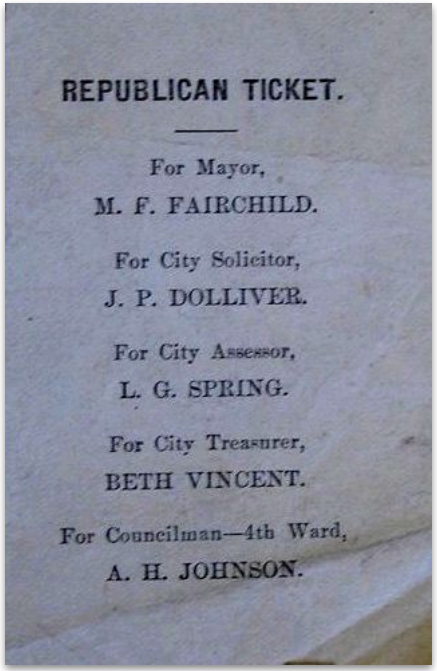


# Voice Voting

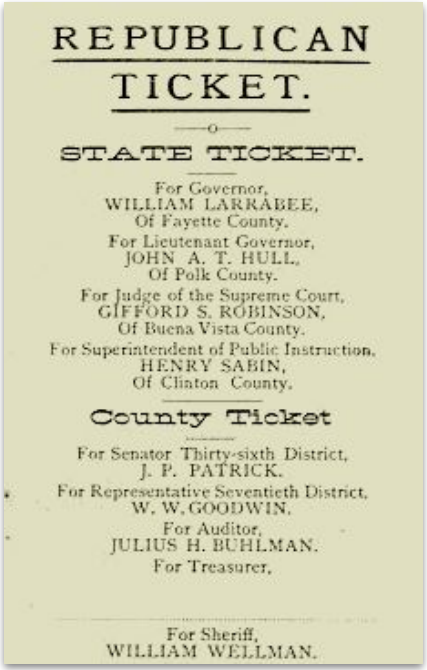
# Voting by Ballots




1839

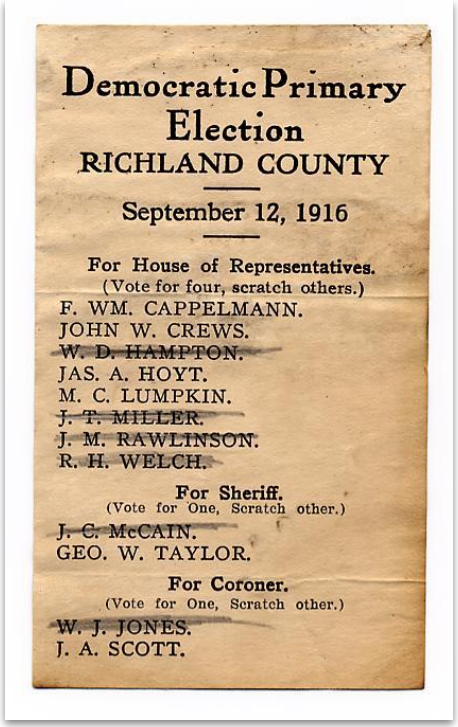



1880




1888

# Voting by Ballots



Democratic Primary Election
RICHLAND COUNTY

September 12, 1916

**For House of Representatives.**
(Vote for four, scratch others.)

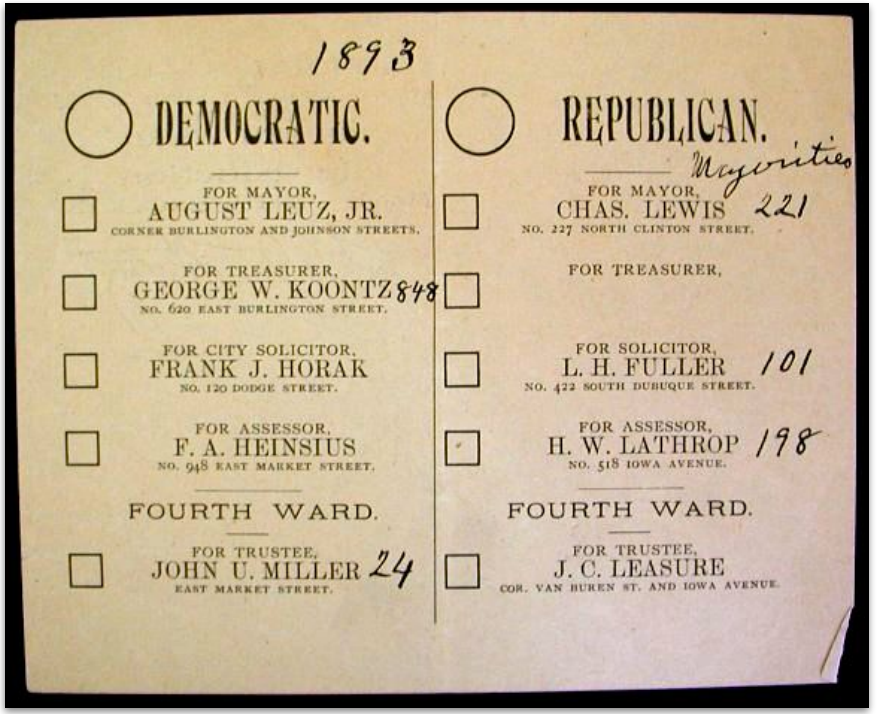
F. WM. CAPPELMANN.
JOHN W. CREWS.
W. D. HAMPTON.
JAS. A. HOYT.
M. C. LUMPKIN.
J. T. MILLER.
J. M. RAWLINSON.
R. H. WELCH.

**For Sheriff.**
(Vote for One, Scratch other.)

J. C. McCAIN.
GEO. W. TAYLOR.

**For Coroner.**
(Vote for One, Scratch other.)

W. J. JONES.
J. A. SCOTT.



1893

DEMOCRATIC.

FOR MAYOR,
AUGUST LEUZ, JR.
CORNER BURLINGTON AND JOHNSON STREETS.

FOR TREASURER,
GEORGE W. KOONTZ 848
NO. 620 EAST BURLINGTON STREET.

FOR CITY SOLICITOR,
FRANK J. HORAK
NO. 120 DODGE STREET.

FOR ASSESSOR,
F. A. HEINSIUS
NO. 948 EAST MARKET STREET.

FOURTH WARD.

FOR TRUSTEE,
JOHN U. MILLER 24
EAST MARKET STREET.

REPUBLICAN.

Majorities

FOR MAYOR,
CHAS. LEWIS 221
NO. 227 NORTH CLINTON STREET.

FOR TREASURER,

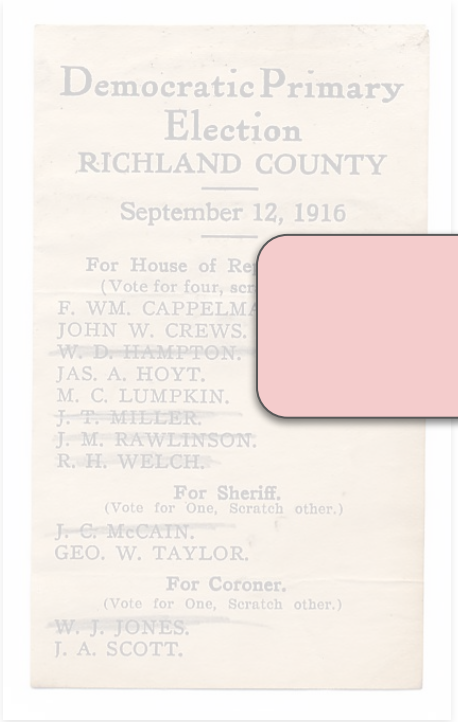
FOR SOLICITOR,
L. H. FULLER 101
NO. 422 SOUTH DUBUQUE STREET.

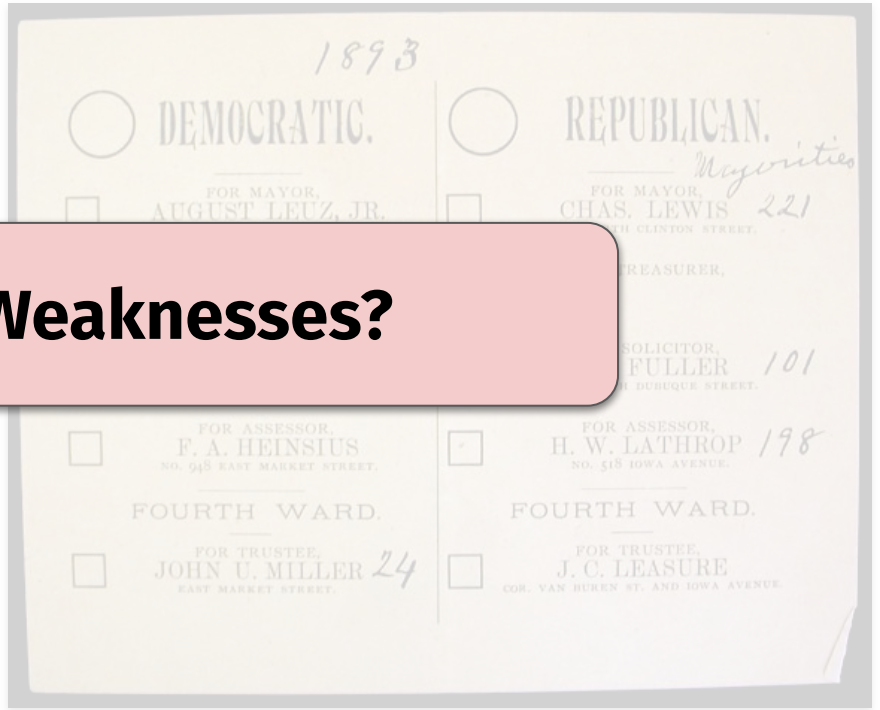
FOR ASSESSOR,
H. W. LATHROP 198
NO. 518 IOWA AVENUE.

FOURTH WARD.

FOR TRUSTEE,
J. C. LEASURE
COR. VAN BUREN ST. AND IOWA AVENUE.

# Voting by Ballots

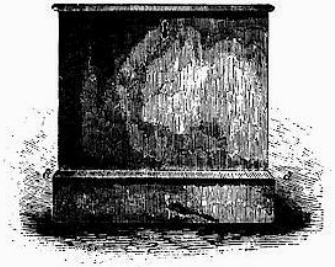



**Weaknesses?**

# Ballot Boxes

# Ballot Boxes



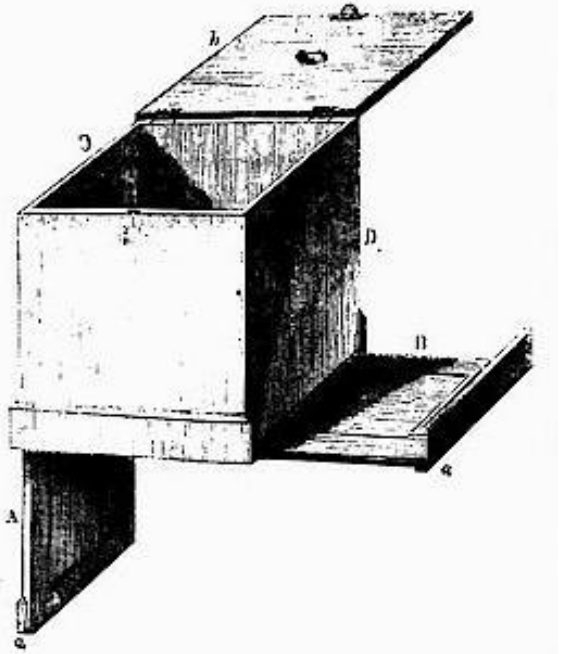
THE STUFFER'S BALLOT-BOX CLOSED UP.

STUFFER'S BALLOT-BOX.

WE give three views of the "Stuffer's Ballot-Box," which will give the reader a clear idea of the *modus operandi* of conducting the elections in San Francisco, and probably in some of our northern cities. The drawings were made from the box now in possession of the Vigilance Committee. It was from ballots taken from this box that Yankee Sullivan made out the election returns that secured Casey his office of Supervisor. The box is about two feet long and fourteen inches wide, and a foot deep, and painted on the outside a dark sky-blue color. It had moulding or cleets around the bottom, and at the top next the lid. The lock, which looked like an ordinary one, is so constructed that though it is worked with a key, it might also be opened by a peculiar pressure upon one side of the lid. There was an auger hole in the middle of the lid, and some of the wax with which it had been sealed at the closing of the polls when last used, was still remaining. It seems that the box was used last at a primary election in the Seventh ward, and the votes were still in it. On looking at the ballot-box, few would suspect the contrivances about it; but on further and minute examination it was found that it had a false bottom and a false side, sliding in grooves, under and behind which were packed quantities of spurious votes all ready for an election.

The mode of working the machine seems to have been this: A sufficient number of the votes which the initiated wished to elect, were prepared and secreted under and behind the false bottom and side. The election was held: Smith was the man to be elected, but Brown was the man of the people's choice. The polls were then closed, and the box sealed and placed in the hands of some one in the secret. The stuffer then drew out the false bottom at his convenience, turned the box upside down, shoved the bottom back and Smith had a majority of the votes: or suppose Brown had still a majority, the false side was pulled down, and another reservoir of votes for Smith was opened. Smith now had a triumphant majority, though the seal had not been touched; or if nothing else would do, a handful of votes for Smith might be easily thrown in, and in each case the lid would probably be opened, and polled votes corresponding with the number of the stuffed ones be withdrawn. One thing was certain—Smith would be elected.

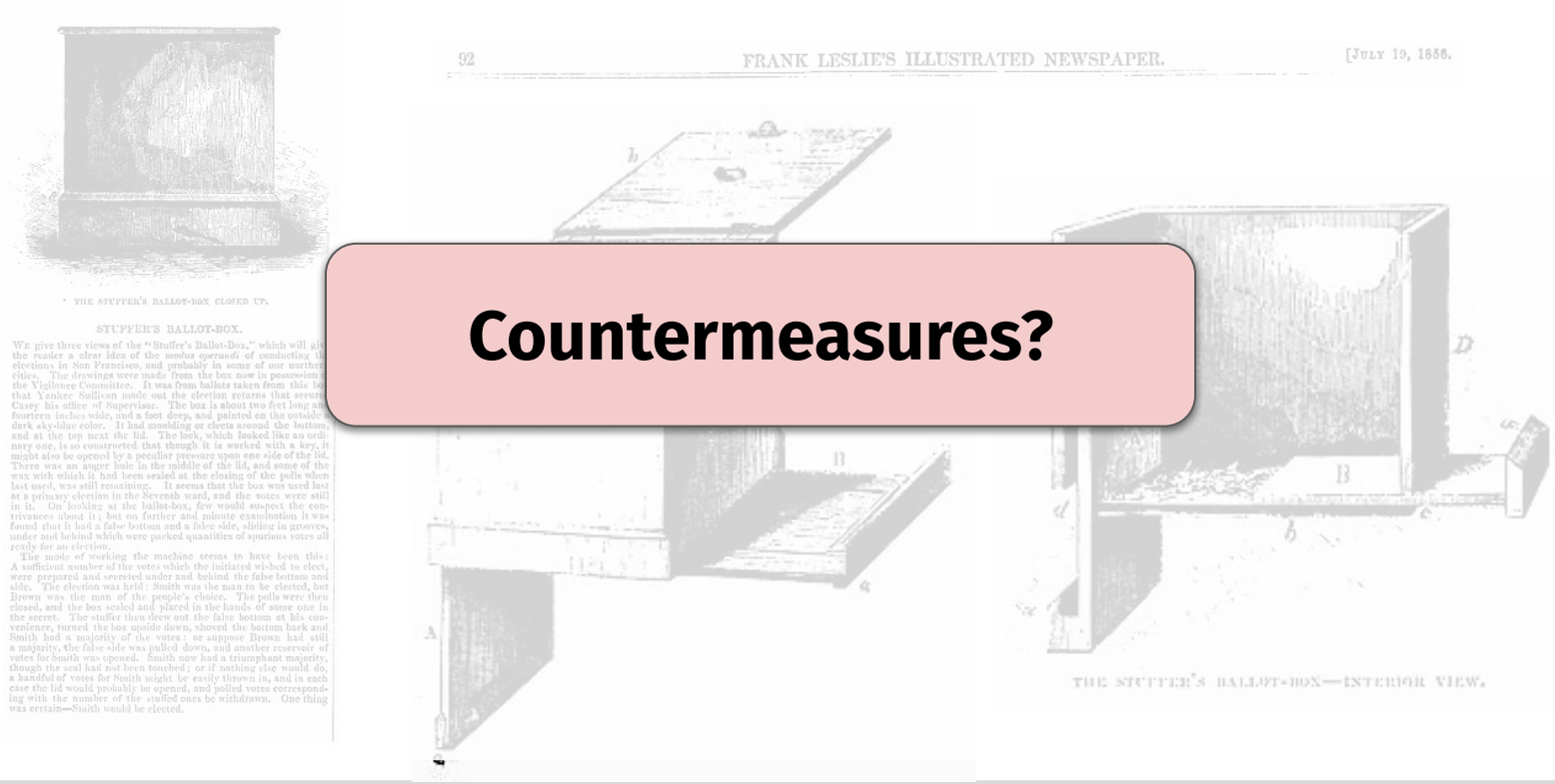
92 FRANK LESLIE'S ILLUSTRATED NEWSPAPER. [JULY 19, 1856.





THE STUFFER'S BALLOT-BOX—INTERIOR VIEW.

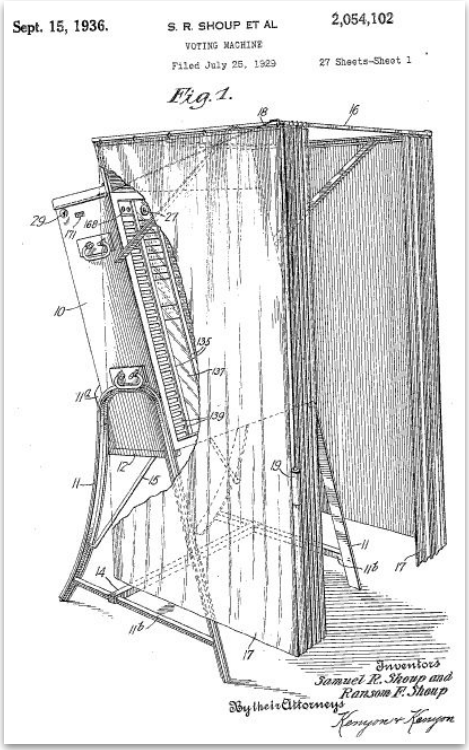
# Ballot Boxes




**Countermeasures?**

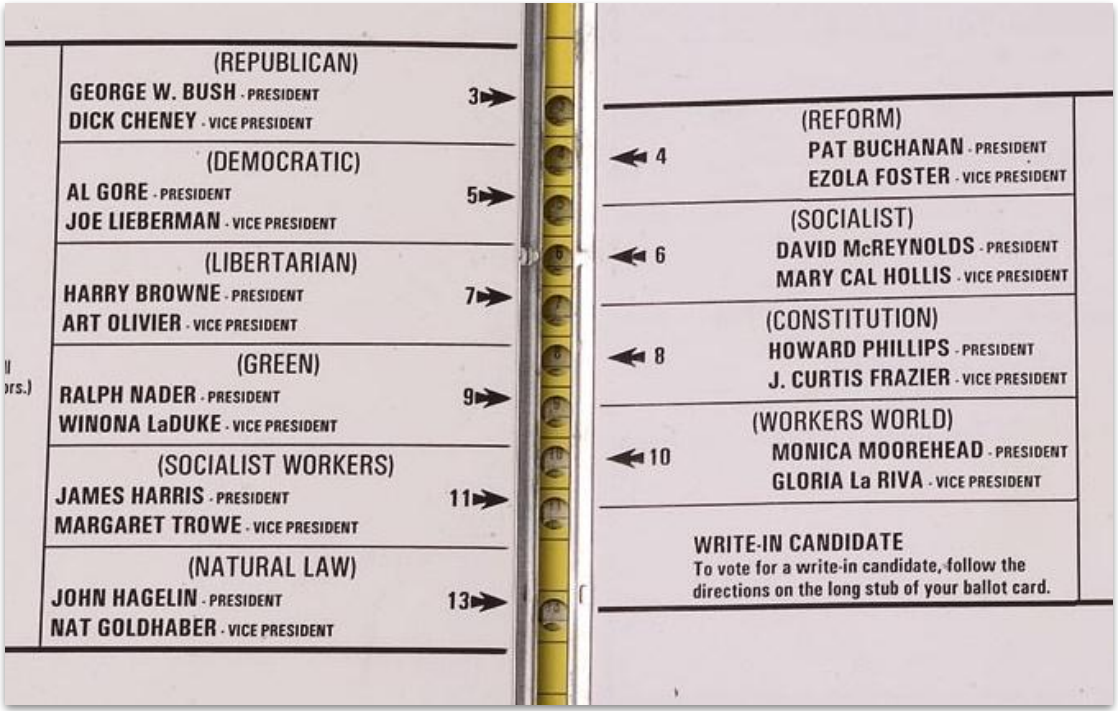
# Ballot Boxes

# Voting Machines

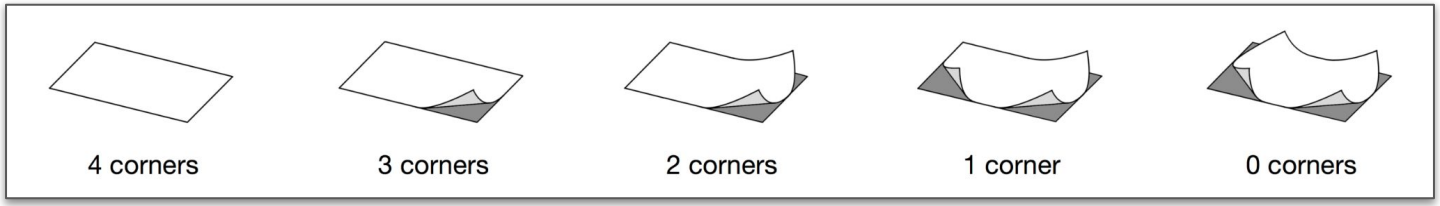

Sept. 15, 1936. S. R. SHOUP ET AL 2,054,102
VOTING MACHINE
Filed July 25, 1929 27 Sheets-Sheet 1
Fig. 1.
Inventors
Samuel R. Shoup and
Ransome F. Shoup
By their Attorneys
Kenyon & Kenyon

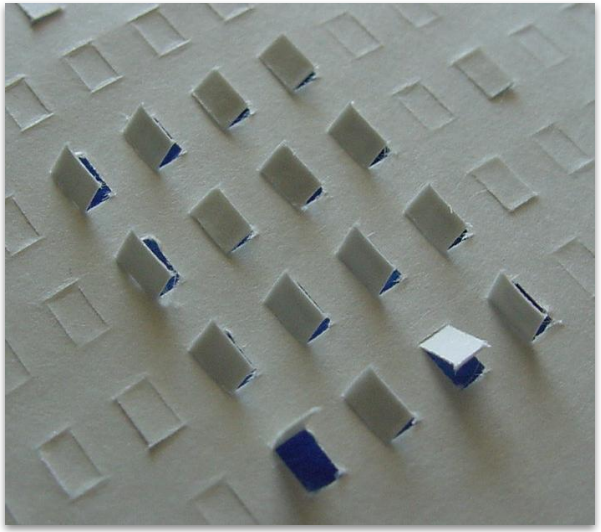
# Voting Machines

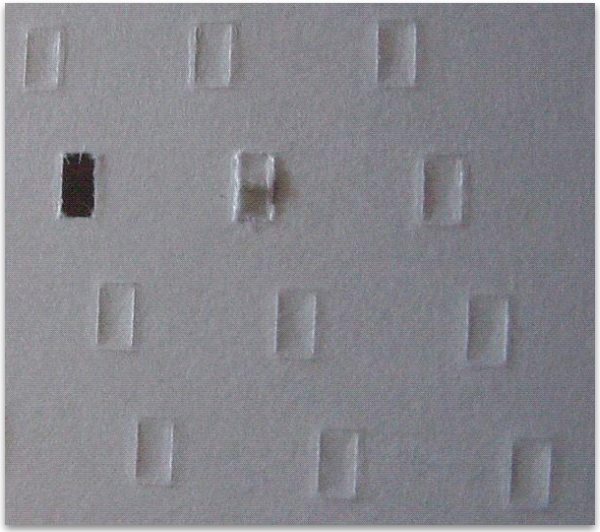

# Voting Machines



# Voting Machines

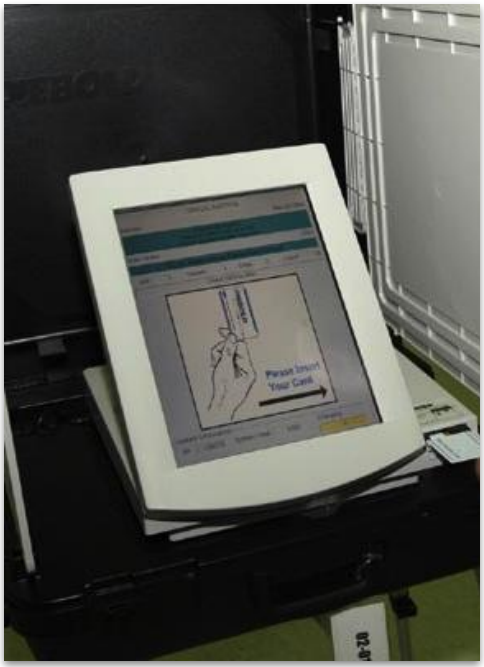




# Computerized Voting

# Early Computer-based Voting



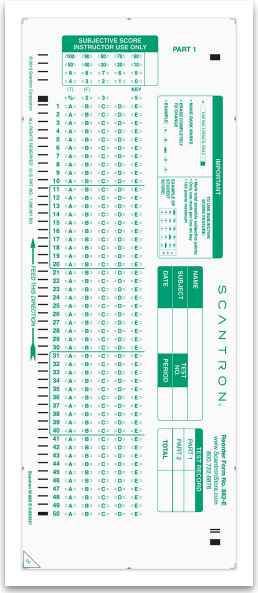

DRE Machine


Optical Scanner

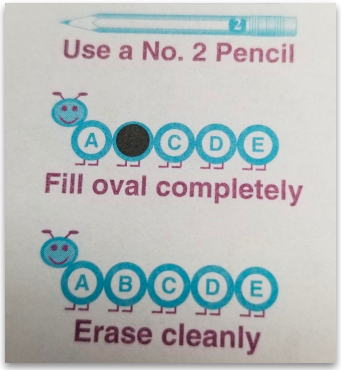
# Optical Scanning

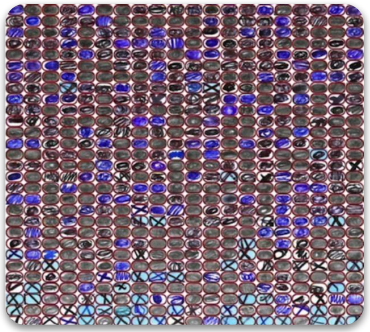

=


+

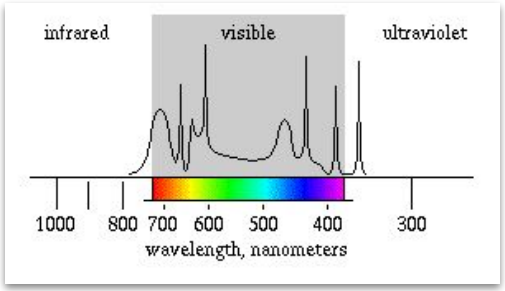


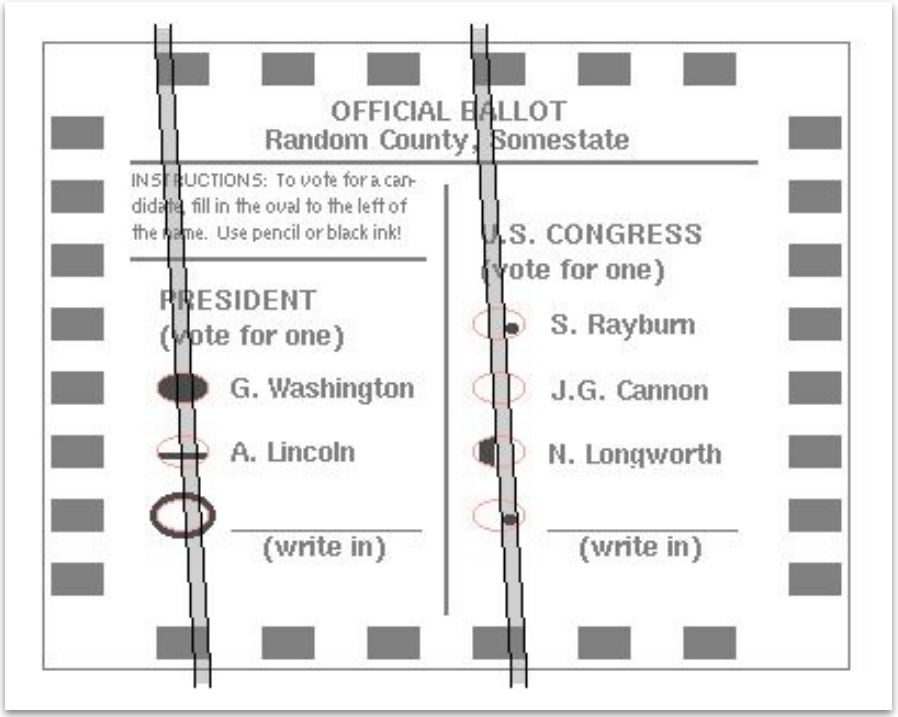
# Optical Scanning



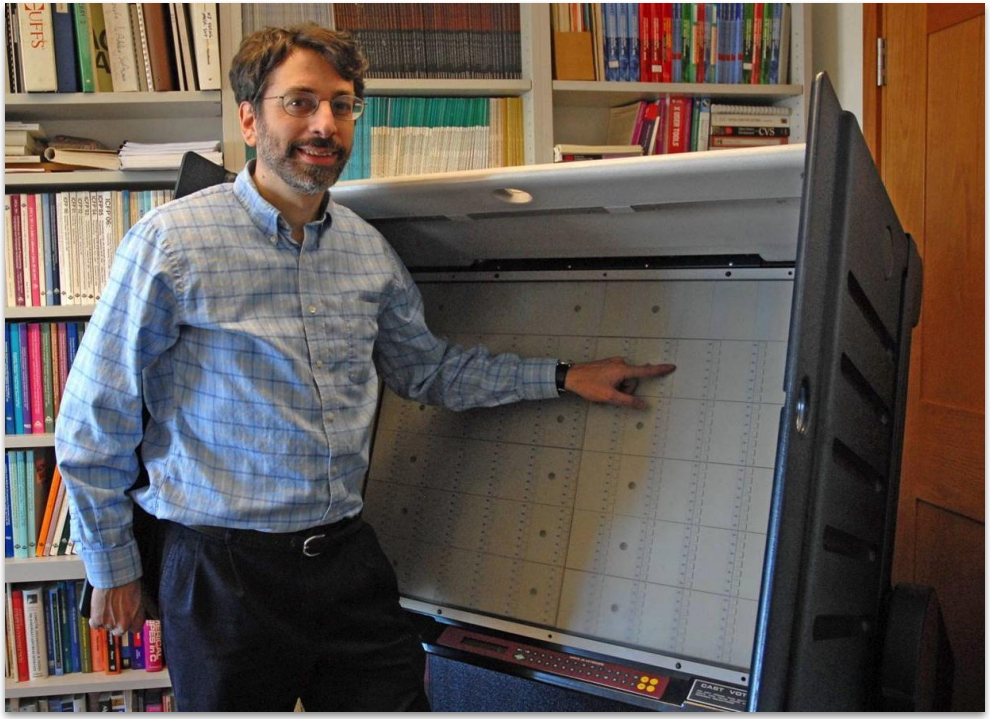
# Optical Scanning

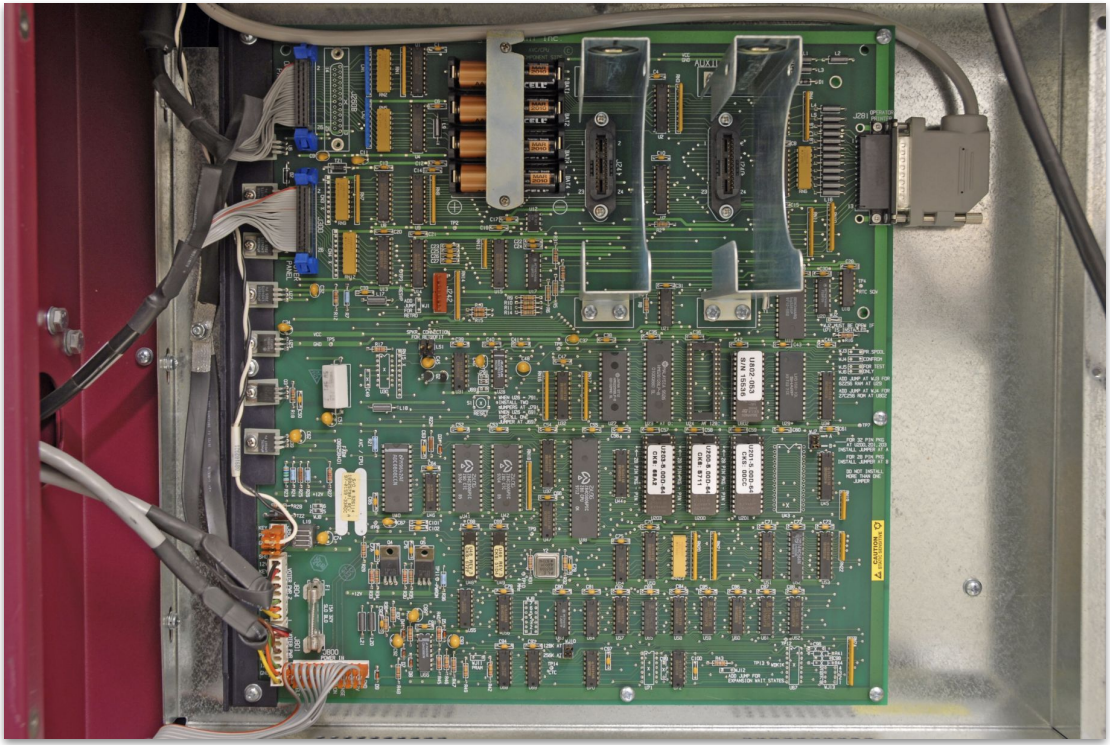



# Attacks against Computerized Voting

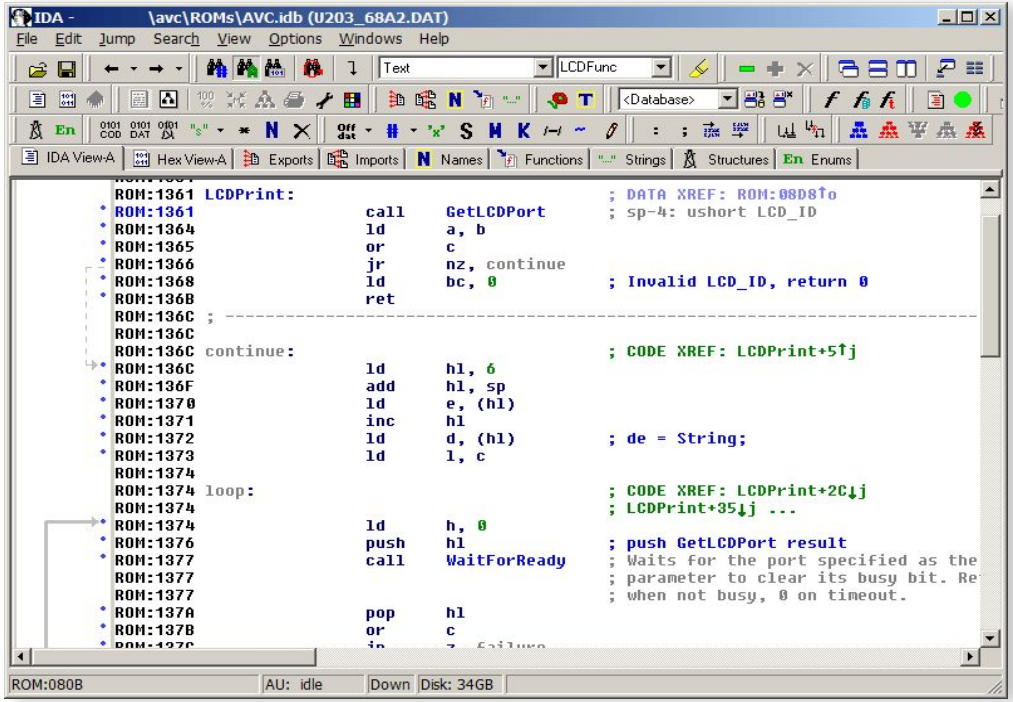
# Sequoia AVC



# Sequoia AVC



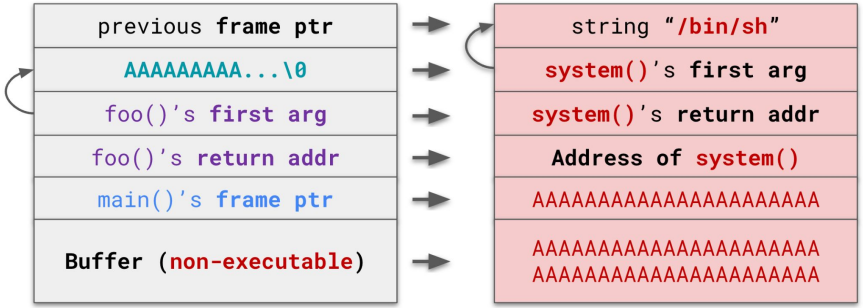
# Attacking the Sequoia AVC

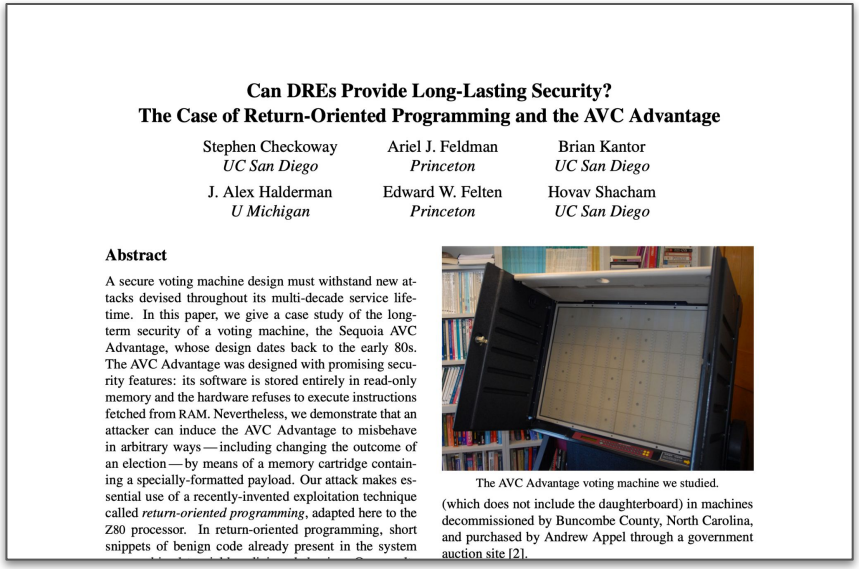


# Attacking the Sequoia AVC

**Return-oriented Programming (ROP)**

Use code gadgets to achieve functionality



**Can DREs Provide Long-Lasting Security?**
**The Case of Return-Oriented Programming and the AVC Advantage**


Stephen Checkoway, *UC San Diego*
Ariel J. Feldman, *Princeton*
Brian Kantor, *UC San Diego*
J. Alex Halderman, *U Michigan*
Edward W. Felten, *Princeton*
Hovav Shacham, *UC San Diego*



**Abstract**

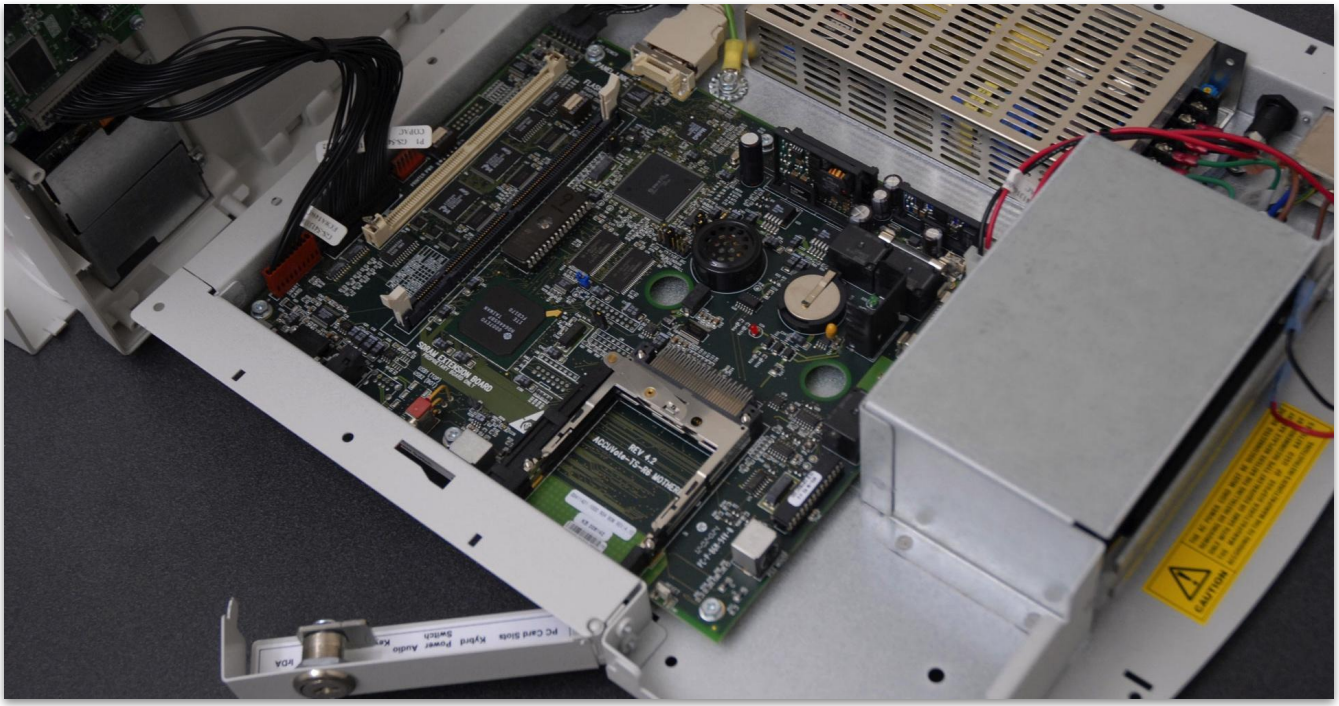
A secure voting machine design must withstand new attacks devised throughout its multi-decade service lifetime. In this paper, we give a case study of the long-term security of a voting machine, the Sequoia AVC Advantage, whose design dates back to the early 80s. The AVC Advantage was designed with promising security features: its software is stored entirely in read-only memory and the hardware refuses to execute instructions fetched from RAM. Nevertheless, we demonstrate that an attacker can induce the AVC Advantage to misbehave in arbitrary ways — including changing the outcome of an election — by means of a memory cartridge containing a specially-formatted payload. Our attack makes essential use of a recently-invented exploitation technique called *return-oriented programming*, adapted here to the Z80 processor. In return-oriented programming, short snippets of benign code already present in the system


The AVC Advantage voting machine we studied.

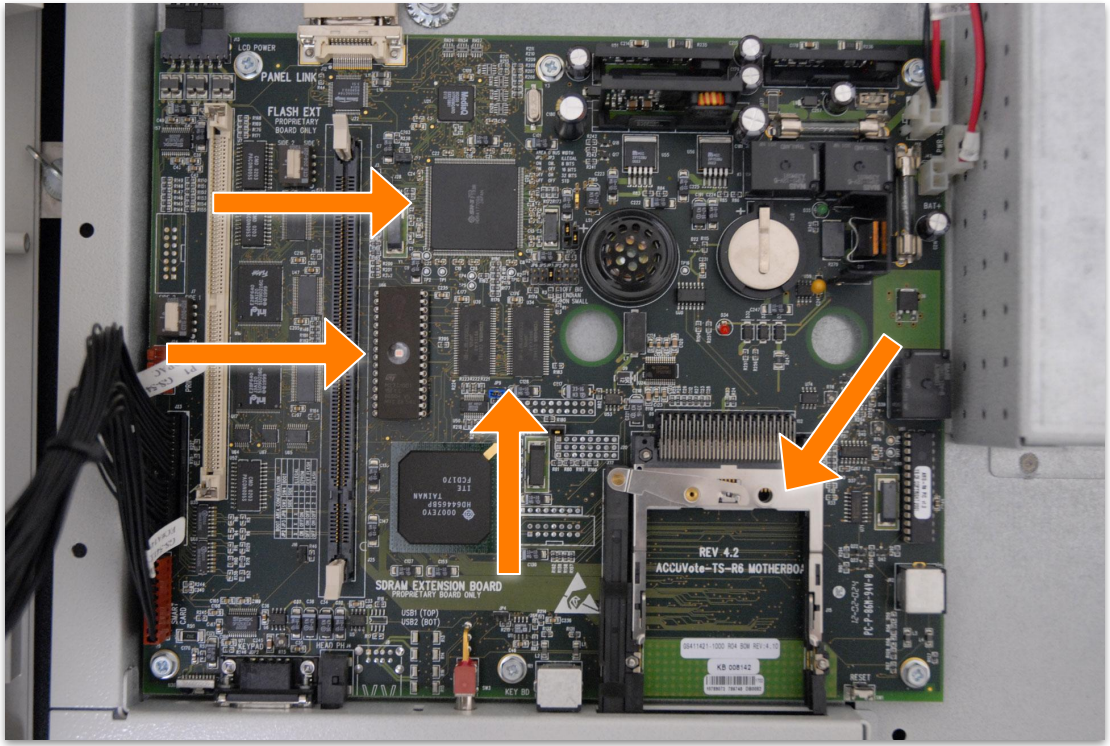
(which does not include the daughterboard) in machines decommissioned by Buncombe County, North Carolina, and purchased by Andrew Appel through a government auction site [2].

# Diebold DRE

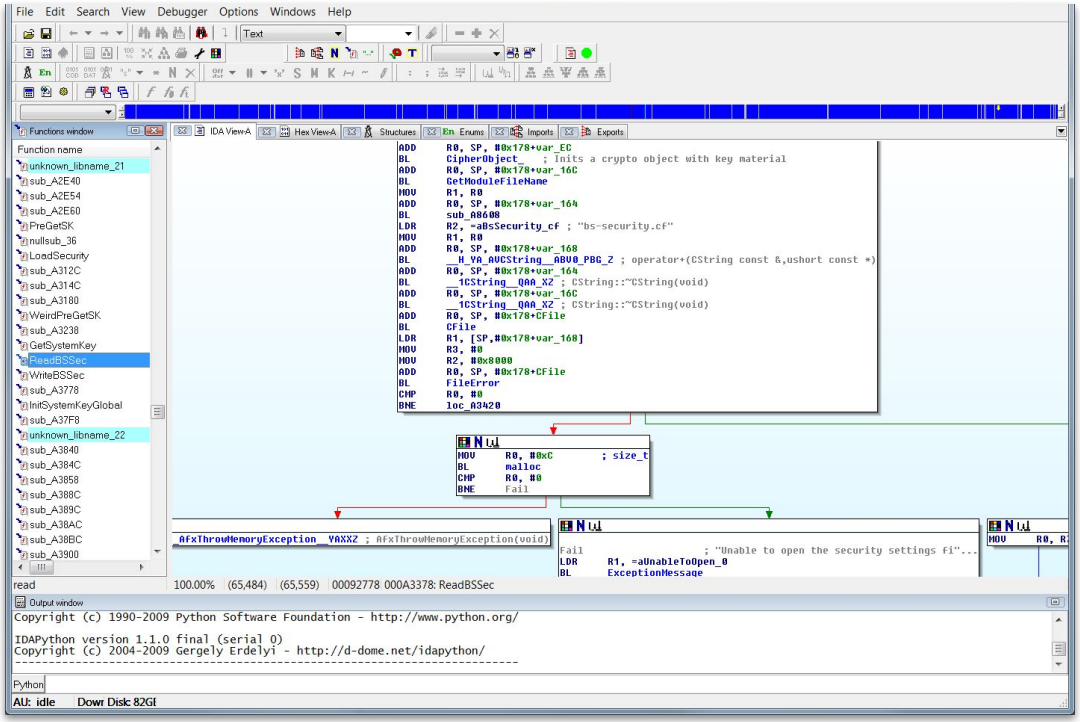
# Reverse Engineering the Diebold DRE



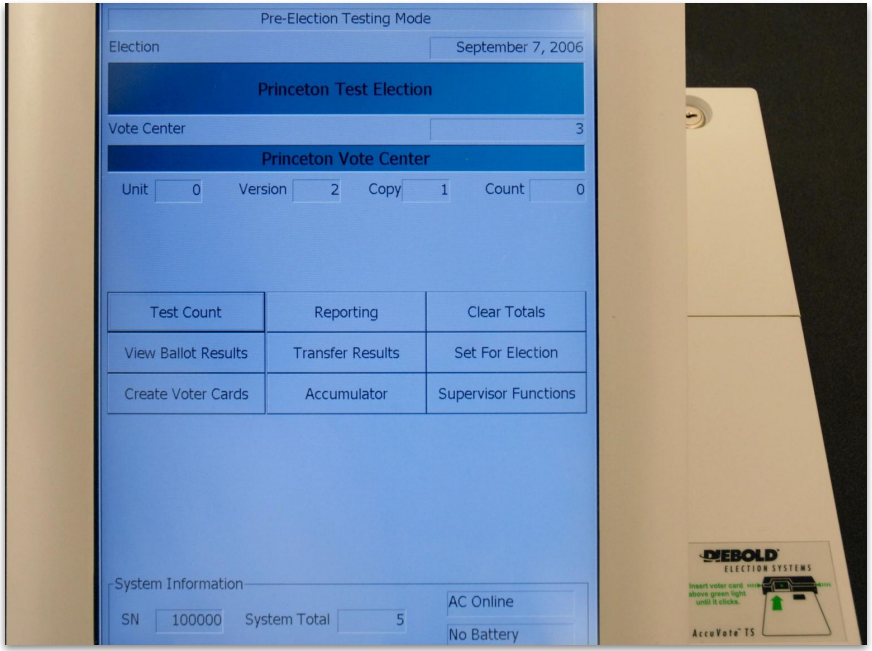
# Reverse Engineering the Diebold DRE

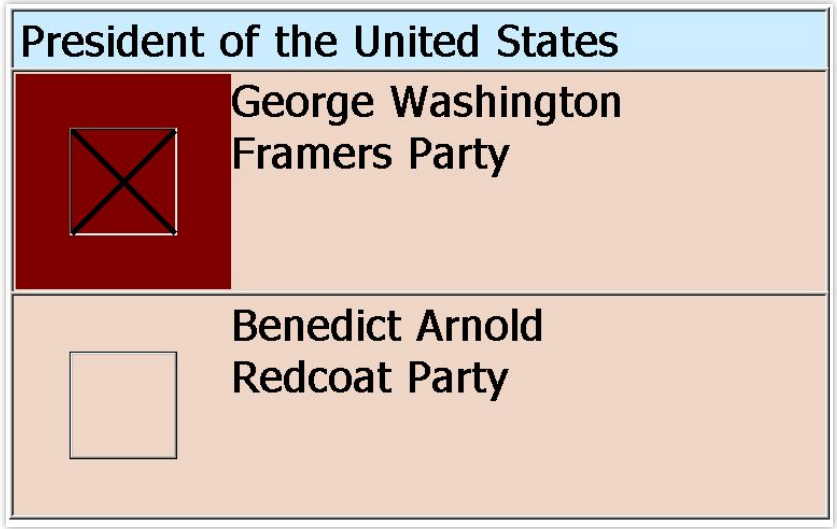


# Reverse Engineering the Diebold DRE

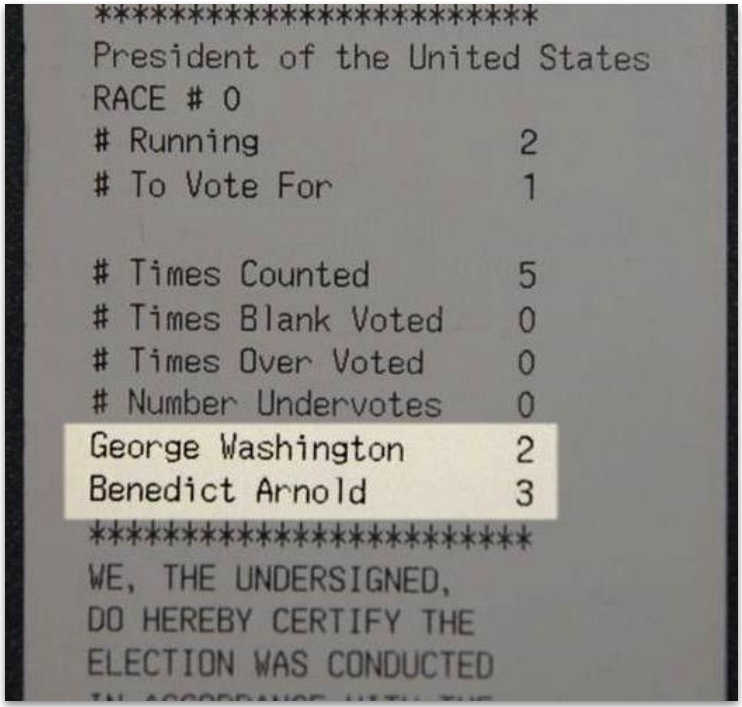


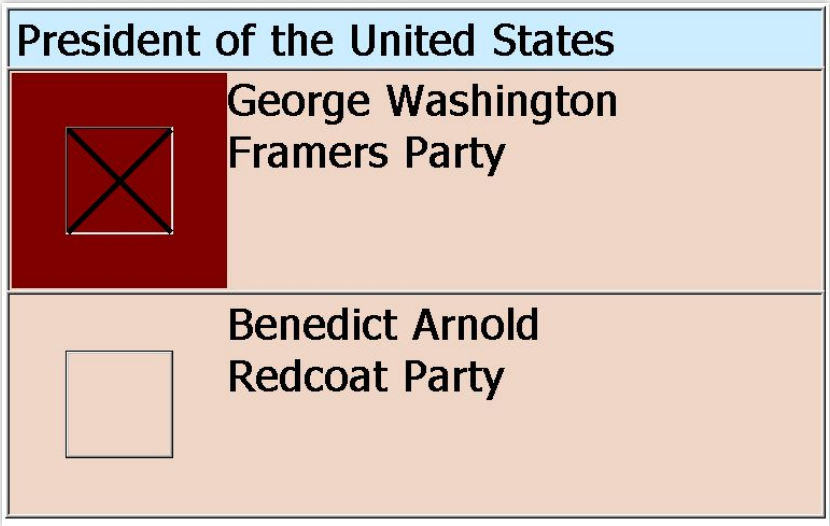
# Attacking the Diebold DRE



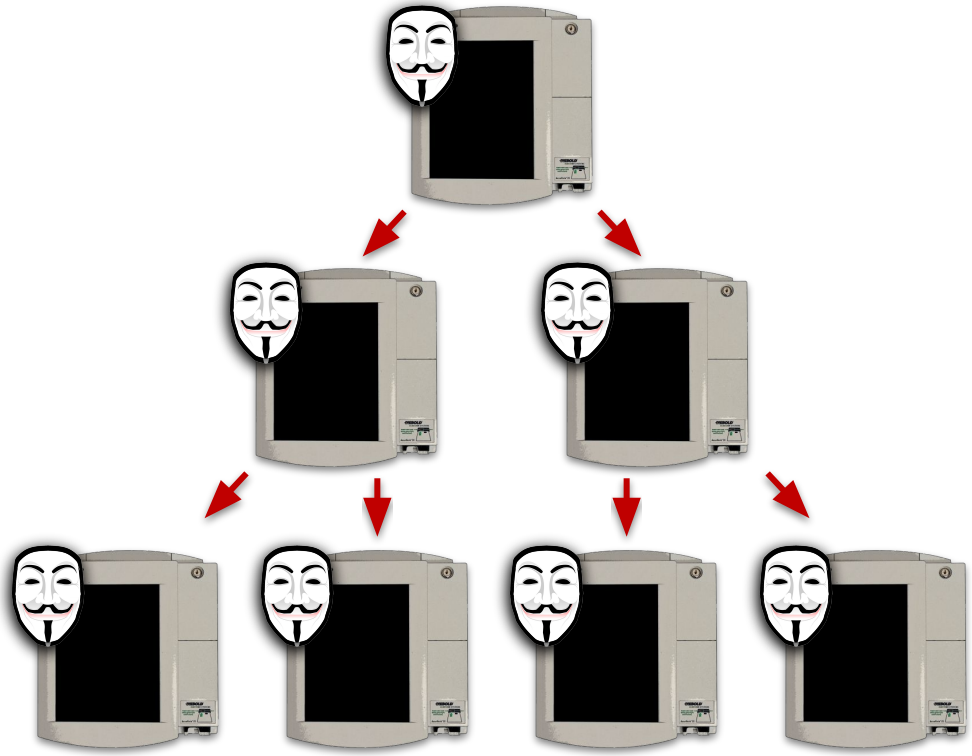



# Attacking the Diebold DRE



# Attacking the Diebold DRE



# Attacking the Diebold DRE

# Attacking the Diebold DRE

# Attacking the Diebold DRE

**Replacement Access Keys**

- 2 keys that allow easy service access to the Tally Printer and replacement battery compartment

GS-567311-1000 **$5.90** USD per set
**$6.90** CAD per set

Enter a quantity

add to your order ▸

ORDER BY PHONE 800.769.3246

# Attacking the Diebold DRE

## Security Analysis of the Diebold AccuVote-TS Voting Machine

Ariel J. Feldman*, J. Alex Halderman*, and Edward W. Felten*,†

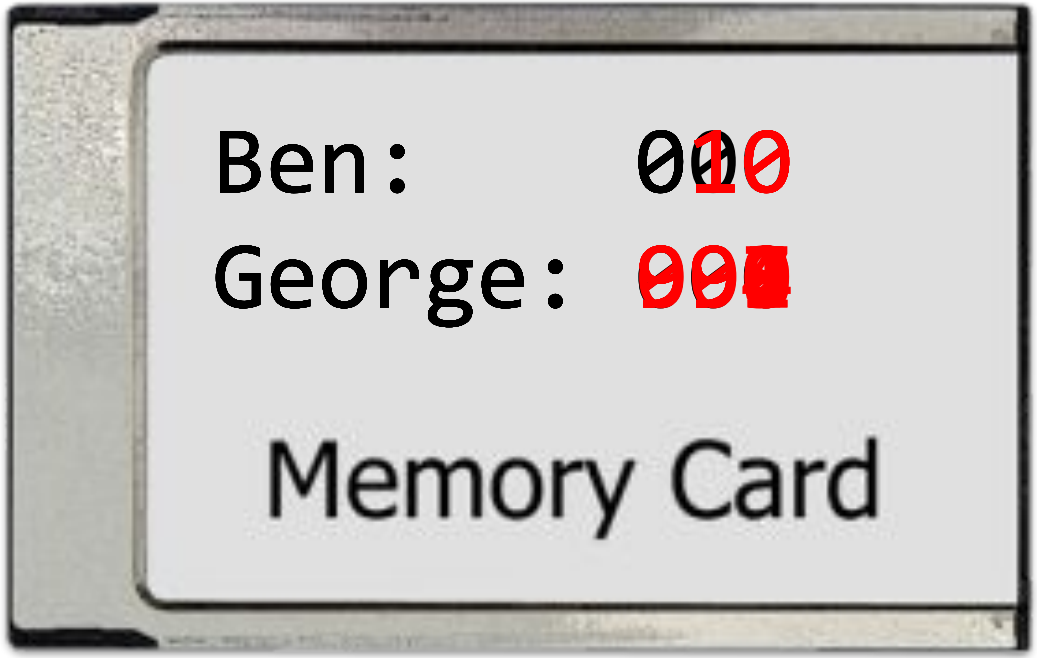
*Center for Information Technology Policy and Dept. of Computer Science, Princeton University
†Woodrow Wilson School of Public and International Affairs, Princeton University
{ajfeldma,jhalderm,felten}@cs.princeton.edu

September 13, 2006

**Abstract**

This paper presents a fully independent security study of a Diebold AccuVote-TS voting machine, including its hardware and software. We obtained the machine from a private party. Analysis of the machine, in light of real election procedures, shows that it is vulnerable to extremely serious attacks. For example, an attacker who gets physical access to a machine or its removable memory card for as little as one minute could install malicious code; malicious code on a machine could steal votes undetectably, modifying all records, logs, and counters to be consistent with the fraudulent vote count it creates. An attacker could also create malicious code that spreads automatically and silently from machine to machine during normal election activities—a voting-machine virus. We have constructed working demonstrations of these attacks in our lab. Mitigating these threats will require changes to the voting machine's hardware and software and the adoption of more rigorous election procedures.

# Attacking the Diebold DRE



# Attacking the Diebold DRE

## Hursti Hack

Add languages

Article Talk | Read Edit View history Tools

From Wikipedia, the free encyclopedia

The **Hursti Hack** was a successful attempt to alter the votes recorded on a Diebold optical scan voting machine. The hack is named after Harri Hursti.

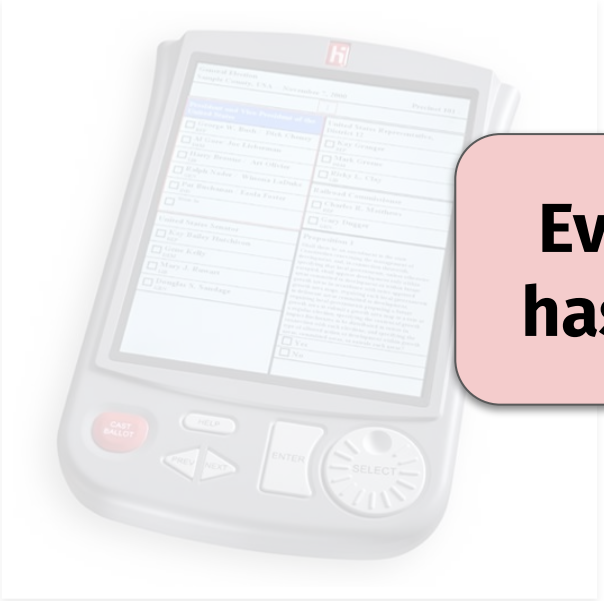
### Participants [ edit ]

The participants were:

- Ion Sancho, Supervisor of Elections, Leon County, Florida.
- Thomas James, Information Systems Officer for Leon County, Florida
- Bev Harris, Black Box Voting founder
- Kathleen Wynne, Black Box Voting Associate Director
- Harri Hursti, computer programmer and security expert
- Hugh Thompson, application security expert and Ph.D. in math
- Susan Bernecker, former Republican candidate for New Orleans city council.
- Susan Pynchon, Director of Florida Fair Elections Coalition

# Other Machines


# Other Machines




**Every examined machine has critical security flaws!**

# Other Machines

LILY HAY NEWMAN SECURITY 08.28.18 11:04 AM

## VOTING MACHINES ARE STILL ABSURDLY VULNERABLE TO ATTACKS

BILL CLARK/GETTY IMAGES

WHILE RUSSIAN INTERFERENCE operations in the 2016 US presidential elections focused on misinformation and targeted hacking, officials have scrambled ever since to shore up the nation's vulnerable election infrastructure. New research, though, shows they haven't done nearly enough, particularly when it comes to voting machines.

## Voting Machine Manual Instructed Election Officials to Use Weak Passwords

**A vendor manual for voting machines used in about ten states shows the vendor instructed customers to use trivial, easy to crack passwords and to re-use the passwords when changing log-in credentials.**

SHARE TWEET

Image: Shutterstodck

States and counties have had two years since the 2016 presidential election to educate themselves about security best practices and to fix security vulnerabilities in their election systems and processes. But despite widespread concerns about election interference from state-sponsored hackers in Russia and elsewhere, apparently not everyone received the memo about security, or read it.

An election security expert who has done risk-assessments in several states since

Latest

The Socialist Memelords Radicalizing Instagram

16 minutes ago

This Guy Wants to Open a DIY Tesla Repair Shop

an hour ago

Scientists Found Antibiotic-Resistant Bacteria In Space

2 hours ago

Supreme Court Weighs Whether Apple's App Store Is a Monopoly

# Internet-based Voting

- **Risks?**

# Risks of internet-based voting?

Nobody has responded yet.

Hang tight! Responses are coming in.

# Internet-based Voting

- **Risks?**

Web Vulnerabilities

Malware

Fraudsters

Denial of Service

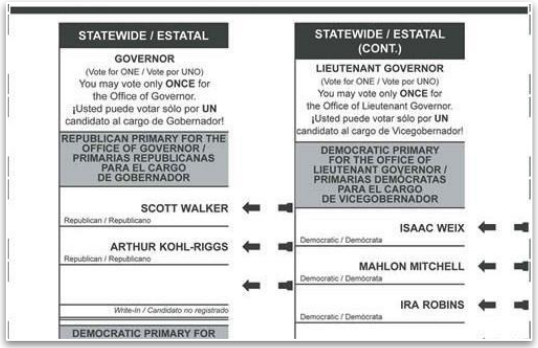
# Internet-based Voting

- Risks?

# Post-election Auditing

# Post-election Auditing

**Redundancy** + **multiple failure modes** = **greater security**

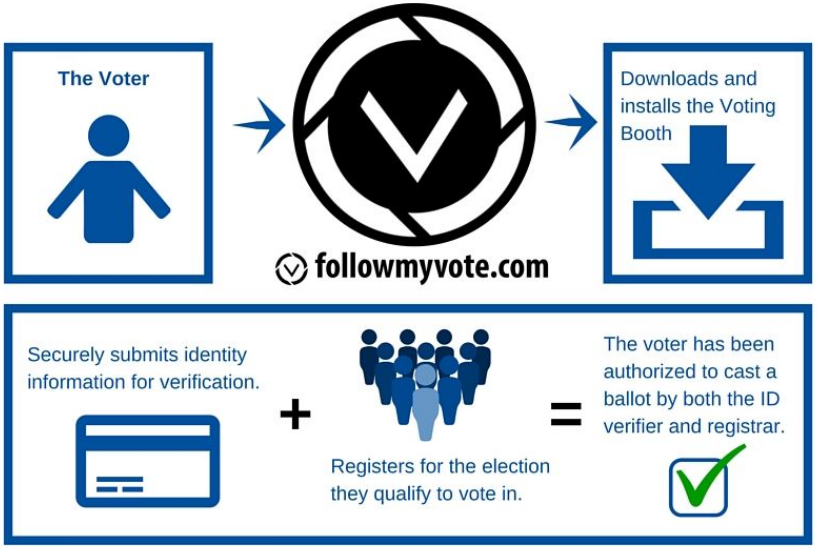


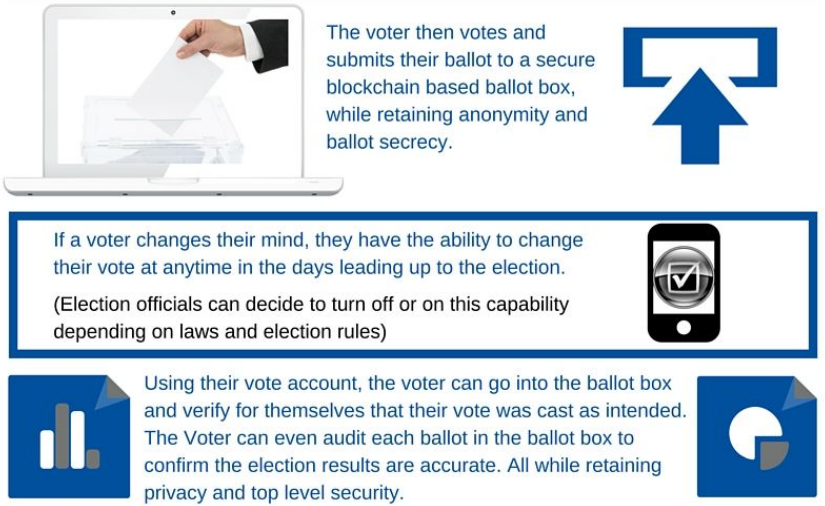


**But... redundancy only helps if we use both records!**

# Post-election Auditing

- **Better ideas?**



# Questions?

# Next time on CS 4440...

Side Channel Attacks & Hardware Security