Mar 23, 2005 -- Lecture 23

THE UNIVERSITY OF IOWA

22C:169 Computer Security

Douglas W. Jones

Department of Computer Science

Transaction Logs

A transaction log records

In chronological order

- *what was changed*
- *when it was changed*
  - Entire database can be reconstructed from log!

Ideally, log is stored

- as a purely sequential file, on
- WORM media such as CD-R
  - Write Once, Read Mostly (Multiple)

Common compromise:

Make periodic backup

- *Daily, weekly, monthly?*
- *Some backups are archival*

Keep transaction logs

- *if all transaction logs are archived*
- *can recover state as of any date!*

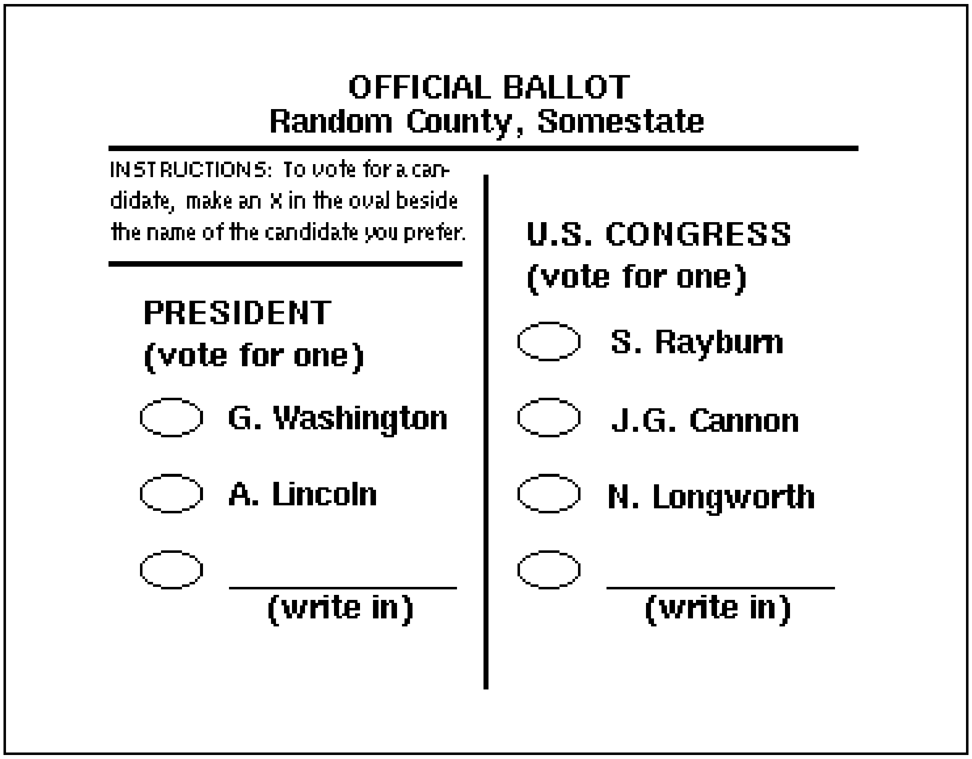
On failure

- *Roll back to most recent backup*
- *Use log to roll forward to point of failure*

Example of Conflicting Requirements

Elections

Database contains all ballots cast



OFFICIAL BALLOT
Random County, Somestate

INSTRUCTIONS: To vote for a candidate, make an X in the oval beside the name of the candidate you prefer.

PRESIDENT
(vote for one)

○ G. Washington

○ A. Lincoln

○ ____________
(write in)

U.S. CONGRESS
(vote for one)

○ S. Rayburn

○ J.G. Cannon

○ N. Longworth

○ ____________
(write in)

Typical Election Requirements

Integrity

    Ballots may not be lost or altered

Privacy

    Nobody may find out how you voted

Secrecy

    You may lie about your vote

Auditability

    It is possible to show that the above constraints were met

Openness

    All election records are public

Integrity, auditability and openness

These are compatible

- *Keep a transaction log*
- *Who cast what ballot when*
- *Publish log and ballots*

Observers can easily determine

- *Who voted when*
- *Compare this with log*
- *Compare log with ballot database*

Privacy and Secrecy

These are compatible

keep no transaction log
randomize ballots in ballot box
publish ballots only after all votes cast

We have a conflict here
Building a voting system that
meets these conflicting demands
is extraordinarily difficult