Civitas
A Secure Remote Voting System

Michael Clarkson

CS 513

November 7, 2007

Joint work with
Andrew Myers and Steve Chong

Coin Crawford 413, ca. 63 B.C., commemorates secret ballot introduced 137 B.C.

Resources

- Technical report with concrete protocols

  `http://www.cs.cornell.edu/projects/civitas`

- Source code to be released

Motivation

History

- Late 1800s: Voting machines debut in US
- 1929: 24 states have bought lever machines
- 1960s: Punch card machines also in use
- 1970s: First major scandals; people figure out very simple hacks to manipulate results reported by machines
- 2000: Bush *v.* Gore Presidential election. Election night (Nov 7—8) Fox News called FL (and thus nation) for Bush. Very soon after, vote total revealed extremely close. Gore first concedes, then "unconcedes". Contested for 36 days until called for Bush.

History

- What went wrong in FL'00?
  - Clear that African American and Jewish voters targeted with attacks meant to confuse or prevent them from voting
  - Problems with recounting the punch-card ballots ("hanging chads")
  - Political games, especially in determining which precincts got recounted and the standards used.
  - US Supreme Court eventually calls off manual recount.
- Outcome:
  - Bush won by 537 votes, only .01% margin of victory. Compare to >3% votes simply lost!
  - Studies since then have concluded that full manual recount would have given Gore victory by > 30,000 votes.

History

- Spring 2001: Sequoia touting its computerized voting system as the solution to the archaic technologies that were the (supposedly) real problem in FL. "Flawless" used as the buzzword. These are touch-screen DREs. In fact, numerous flaws occurred---votes lost, machines that wouldn't boot for hours, machines that froze when certain candidates were selected, etc.
- Nov 2002: FL meltdown #2. Bush pushes through HAVA, giving $4B in aid to states to purchase new voting machines. Primary vendors are Diebold, ES&S, Sequoia, and minor vendor is Hart. Many states bought these, but situation never improved.

History

- 2003: Bev Harris discovered source code for Diebold system. Several computer scientists publish a paper shredding the system. This depite "certification" by federal labs.
- From 2003-7: The same story goes on and on. Industry keeps assuring the public that the machines are "secure". Academics and activists continue to point out the security risks.
- Many call for VVPAT
  - In principal, audit is good!
  - But recounts can be manipulated, not sure that electronic copy is identical to paper. And is paper really that sacred? The Fed moves bits, not cash.

History

- March 2007: California Secretary of State Debra Bowen commissions "Top-to-bottom Reviews" of three major commercial voting systems.
- August 2007: Results released.
  - "The security of elections…depends almost entirely on the effectiveness of election procedures." [Wagner et al.]
  - "An attacker could subvert a single polling place device…then …reprogram every polling place device in the county." [Wallach et al.]
  - "…numerous programming errors, many of which have a high potential to introduce or exacerbate security weaknesses…In every case we examined the cryptography is easily circumvented." [Blaze et al.]
- Bowen decertifies all these systems, recertifies with "procedural constraints"

Security Properties

Remote Voting

- *Remote*: no supervision of voting or voters
  - Generalizes Internet voting
- The right problem to solve:
  - Does not assume supervision
  - Internet voting is common (Debian, ACM, SERVE)
  - Absentee ballots
  - Some states moving entirely to remote voting (Oregon, Washington)

Integrity

- The final tally cannot be changed to be different from a public counting of the votes

Verifiability:

The final tally is verifiably correct

- Including:
  - Voters can check their own vote is included (*voter verifiability*)
  - Anyone can check that only authorized votes are counted, and no votes are changed during tallying (*universal verifiability*)
  - No ballot stuffing
- Sorely lacking in real-world systems

Confidentiality

- No adversary can learn any more about votes than is revealed by the final tally

Anonymity:

> The adversary cannot learn how voters vote, unless voters collude with the adversary.

- Anonymity is too weak
  - Allows vote selling and coercion of voters

Confidentiality [2]

Coercion resistance:

> Voters cannot prove whether or how they voted, even if they can interact with the adversary while voting.

- Required by Civitas
- Stronger than anonymity (or *receipt-freeness*)
  - Adversary could be your employer, spouse, …
  - Must defend against forced abstention or randomization

Availability

- Unavailability of votes can compromise integrity
  - Missing votes not universally detectable
  - So need to guarantee availability of votes
- Otherwise, availability not guaranteed
  - Software systems implementing authorities
  - Results of election
- Orthogonal extensions possible
  - Byzantine fault tolerance
  - Threshold cryptography

Threat Model

Election Authorities

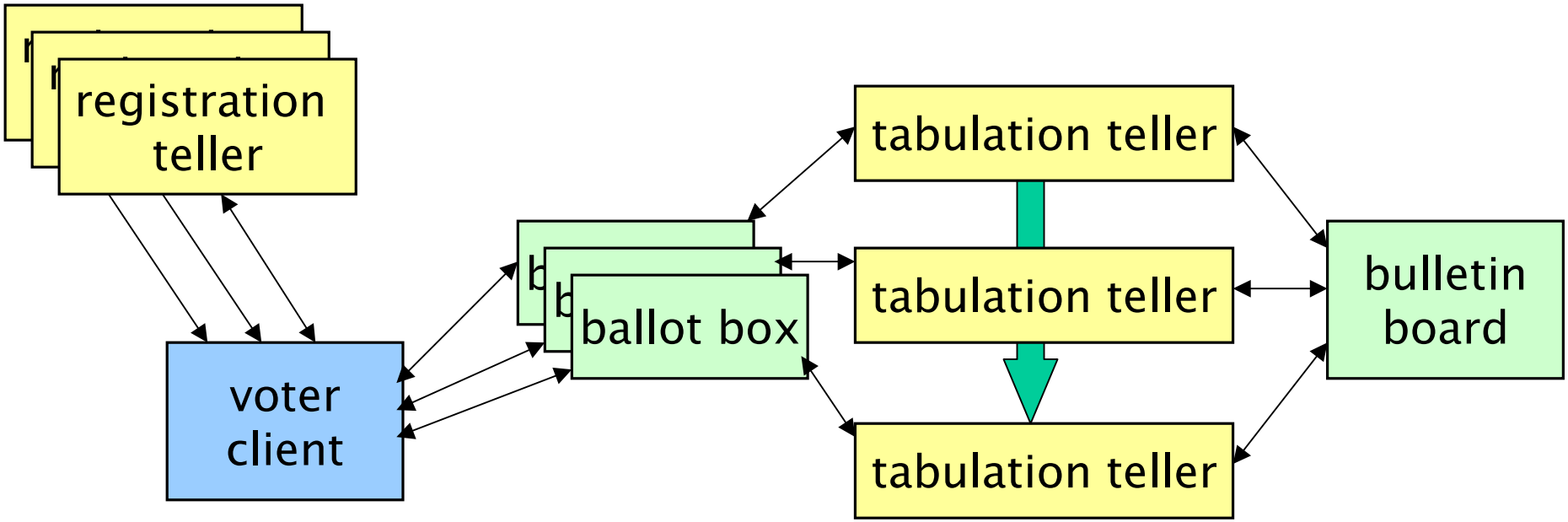
- Agents providing components of the election system
- Various types of authorities
- We assume that all but one (of each type) is malicious

Adversary Model

- May corrupt all but one of each type of election authority
- May coerce voters, demanding any secrets or behavior, remotely or physically
- May control network (Dolev-Yao)

Architecture

Civitas Architecture



Based on voting scheme due to Juels, Catalano, and Jakobsson (2005).

Terminology

- *Voting system:* (software) implementation
- *Voting scheme:* cryptographic construction
- *Voting method:* algorithm for choosing between candidates

<Voting Schemes>

Classification based on cryptographic technique used to achieve confidentiality.

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.
- Many 513 projects based on FOO'92

Mix Networks

- Original Chaumian decryption mix:
  - Implemented with set of servers
  - Input: list of encrypted values
    - Enc(Enc(Enc(…c…)))
  - Output: same list, decrypted
    - But order of list permuted
  - Each server in mix permutes list and removes one layer of encryption
- Problem: servers trusted to be honest
  - Need *robustness*
- Civitas based on mix network
  - Actually uses *reencryption* mix

Mix Networks

- Voting scheme:
  - Voter encrypts vote, authenticates to Ballot Box server, submits vote.
  - Set of tabulation tellers run a mixnet over the encrypted votes, resulting in random permutation of votes.
  - Permuted list is decrypted and tallied.

Homomorphic Encryption

- A *homomorphic* encryption scheme has an operator ★ such that Enc(m) ★ Enc(n) = Enc(m ★ n). ★ is usually either + or ×, never both.
  - E.g. both RSA and El Gamal have ×.
- 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.

</Voting Schemes>

Setup Phase

- Supervisor posts public keys for registrar, registration tellers, tabulation tellers
- Registrar posts identities and public keys of voters
- Tabulation tellers generate public key K_{TT} for a distributed cryptosystem
  - Everyone knows public key
  - Each teller has share of private key
  - Anyone can encrypt; participation of all tellers required to decrypt

Setup Phase [2]

- Registration tellers generate credentials for voters
  - Each credential is a pair of public/private values
  - Each tellers responsible for generating one share of the full credential for each voter
  - Public credential share posted on bulletin board
  - Private credential share stored; later released to voter

Voting Phase

- Voter retrieves private credential shares
  - Contacts each registration teller, authenticates with public key posted by registrar
  - Voter combines all shares to produce full private credential
- Voter votes
  - Submits a copy of vote to each ballot box:

    Enc(private credential; KTT), Enc(choice; KTT), P

  - P is a proof that vote is well-formed

Tabulation Phase

Tabulation tellers:

1. Retrieve data
2. Verify vote proofs
3. Eliminate votes with duplicate credentials
4. Anonymize votes and credentials
5. Eliminate votes with unauthorized credentials
6. Decrypt choices in remaining votes

Tellers constantly post proofs that they are performing the protocols correctly; yields verifiability

Voters Lie to Resist Coercion

- Voter picks random "fake" private credential share
- Constructs new "fake" private credential
- Uses "fake" credential to behave exactly as coercer demands
  - Give it to adversary
  - Submit any vote demanded by adversary
  - Voter needs some time not under coercer's control to use his real credential
- Yields coercion resistance

Cryptographic Techniques

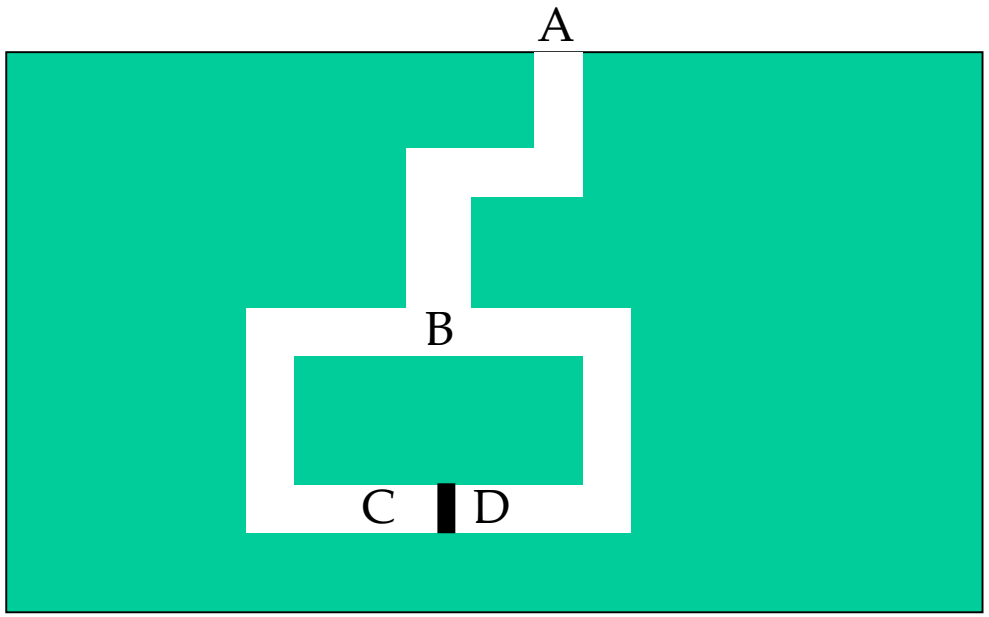
- Zero-knowledge (ZK) proofs
  - Vote proofs, tabulation proofs
- Plaintext equivalence test
  - Elimination of duplicate and unauthorized credentials
- Mix network (already discussed)
  - Anonymization

Zero-Knowledge Proofs

- Standard proofs in math class: student (prover) writes something that TA (verifier) checks.
  - Convinces verifier of statement made by prover
- But standard proofs also reveal knowledge to the verifier
  - Prover: "I know the password to the Federal Reserve"
  - Verifier: "I don't believe you!"
  - Prover: "It's XYZZY".
  - Verifier: Logs into Fed to check if password works.
  - Verifier: "Thanks. Now I know it too."

Zero-Knowledge Proofs

- A *zero-knowledge* proof allows P to prove to V that a statement is true without revealing any more information.
- Example: The magic word and the cave



Zero-Knowledge Proofs

- Commit:
  - V stands at A
  - P walks into cave to either C or D
  - V walks to B
- Challenge:
  - V shouts to P to either come out the left or the right passage
- Response:
  - P complies, using the magic word if necessary
- P and V repeat until V is convinced P knows the magic word

Zero-Knowledge Proofs

- Zero-knowledge:
  - V never learns the magic word
  - V can't convince anyone else that P knows the magic word
    - P & V could have agreed on series of "left/right" in advance
- Large classes of problems have ZK proofs
- This proof was *interactive*
  - Based on challenge/response
  - Can make *noninteractive* by using a special kind of hash function to generate the challenge

Plaintext Equivalence Test

- Special kind of ZK proof
- Tabulation tellers prove (as a group) that Dec(c) = Dec(c') without anyone, including the tellers, learning what Dec(c) or Dec(c') actually are

Security Assurance

Design Proved Secure

- Civitas voting scheme provably achieves coercion resistance and verifiability
- Style of proof:
  - Cryptographic reduction proof
  - Prove that any attack on system would also enable an attack on something more basic
  - In this case, reduces to DDH (El Gamal assumption)
- Proof (and thus security) based on many assumptions

Trust Assumptions

1. DDH, RSA.
2. The adversary cannot masquerade as a voter during registration.
3. At least one RT, TT and ballot box is honest.
4. Each voter has an untappable channel to a trusted registration teller.
5. Voters trust their voting client.
6. The channels from the voter to the ballot boxes are anonymous.

Implementation Verified Secure

- Implemented in Jif
  - Jif = Java + Information Flow [Myers '99, Chong and Myers '04 '05]
  - Security-typed language
  - Static-type checking and dynamic enforcement of information-flow policies
  - Java and C used for some low-level code (esp. cryptography and I/O)
- Jif type checking yields assurance
  - Code is correct with respect to policies
  - Policies can be audited and certified
- Implementation
  - 22,000 LOC in Jif, Java, and C

Performance

Blocking

- Assign voters into blocks
  - Block is a “virtual precinct” or “absentee ballot group”
  - Coercion resistance guaranteed within a block
  - Each block tallied independently of other blocks, even in parallel
  - Registrar implements policy on assignment; best policy might be uniform random
  - Reasonable block size? We use 100.
- Tabulation time is:
  - Quadratic in block size (duplicate/invalid elimination is the culprit)
  - Linear in number of voters, if using one set of machines for many blocks
  - Constant in number of voters, if using one set of machines per block

Experimental Study

- Performance study
  - Ran on 3 GHz CPUs for tab. tellers
  - Keys: (224,2048) El Gamal, 2048 RSA, 128 AES
- Results:
  - Tally block of 100 voters in 1 hour, or 200 voters in 5 hours
  - For K=100, election tallied in 39 sec / voter / authority
  - If CPU costs $1 / CPU / hour, then marginal cost is about 4¢ / voter
  - Cf. current election costs of $1-$3 / voter [International Foundation for Election Systems]

Ranked Voting

Ranked Voting

- Voters submit ranking of candidates
  - E.g. Condorcet, Borda, STV
  - Help avoid spoiler effects
  - Defend against strategic voting
- Tricky because rankings can be used to signal identity ("Italian attack")

- Civitas implements coercion-resistant Condorcet, as well as *approval* and *plurality* voting methods