

Coercion-Resistant Remote Voting Using Decryption Mixes

Michael Clarkson and Andrew Myers
Cornell University

Frontiers in Electronic Elections
September 16, 2005

Remote Voting

- Clear interest
 - SERVE, Debian (devotee), program committees, etc.
- CIVS: Condorcet Internet Voting Service
 - <http://www5.cs.cornell.edu/~andru/civs/>
 - Offers security guarantees:
 - Whether/how a voter votes remains secret, even if server storage compromised
 - *But assuming trusted server software, and without verifiability*
 - Users have run ~100 elections with 10–1700 voters
- Redesign to get **verifiability** and **coercion resistance** without a trusted server

Clarkson and Myers: Coercion-Resistant Remote Voting

2

Trust Model

- We **have** to trust client software
 - Implementation of CIVS2 in Jif
 - Java + Information Flow
 - Check that information flows obey confidentiality and integrity policies
- Move rest of trust into:
 - Cryptography
 - Anonymous channel
 - Set of tellers

Clarkson and Myers: Coercion-Resistant Remote Voting

3

Prêt à Voter (PAV)

- [Chaum, Ryan, Schneider, {4, 0} days ago]
- Uses decryption mix and auditing to remove trust in much of mechanism
- But designed for *supervised voting*, not remote
 - Authentication and handling of ballots rely on trusted officials, booth, and machine

Problem 1: Adapt PAV to Internet voting

Clarkson and Myers: Coercion-Resistant Remote Voting

4

Ranked Voting Methods

- Voters submit ordering of candidates:

Vanilla	4
Chocolate	1
Strawberry	3
Cookie dough	2
Mint chocolate chip	5

- Captures more information about “the will of the people” than binary voting methods
- Condorcet, STV/IRV, Borda, ...

Clarkson and Myers: Coercion-Resistant Remote Voting

5

Covert Channel in Rankings

- Low-order rankings create a covert channel
 - Voter can encode identity using channel

Vanilla	X
Chocolate	X
Strawberry	X
Cookie dough	X
Mint chocolate chip	1

} 4! completions

- Coercion intrinsically possible
 - Many ranked methods require access to the individual votes cast
 - Most schemes, including PAV, make the votes public

Clarkson and Myers: Coercion-Resistant Remote Voting

6

Condorcet Methods

- Benefits:
 - Usually do not require individual ballots
 - Many argue they produce superior results (at least over FPTP)
- *Condorcet winner* (CW) is the candidate who would defeat every other candidate in a one-on-one plurality vote
 - Chocolate beats Vanilla 60-40
 - Chocolate beats Mint 90-10
 } Chocolate is CW
- Resistant to strategic voting
 - Voters have strong incentive to vote true preferences

Problem 2: Adapt PAV to Condorcet methods

Overview

Problem 1: Adapt PAV to Internet voting

- Eliminate trusted supervision
 - Ballot distribution
 - Authentication

Problem 2: Adapt PAV to Condorcet methods

- Eliminate covert channel in ranked ballots

Condorcet Ballots

- Simple: Decompose rankings into a $C \times C$ binary matrix
 - C = number of candidates
 - Cell (i,j) = 1 if voter prefers candidate i to j , 0 otherwise
- Treat each cell as a separate vote
 - Each with its own unique ballot and onion
 - Voter casts $O(C^2)$ 0/1 votes
- Engineering ballot forms
 - No longer PAV's cyclic ordering of fixed set of candidates
 - Let $\text{onion}(D)$ be an onion with innermost layer D
 - Ballot for i vs. j has $\text{onion}(i,j)$
 - Audit sets of ballots for well-formedness

S,M	
	g3R04

Tallying Ballots



Tallying Ballots

- Compute a sum matrix from final column of mix
 - Run any *additive/summable* algorithm for CW
- Coercion resistant:
 - Identifying low-order preferences requires identifying the set of votes from a voter
 - But PAV's decryption mix anonymizes *each* vote in final column
 - Sets not identifiable, so neither are low-order preferences

Ballot Handling

- Problem: LHS+onion of ballot reveals too much
 - Must prevent everyone (except voter) from learning map from LHS to onion
 - Distributor(s) of ballots
 - Creator(s) of ballots
- Our solution: conceal LHS, reveal only to voter

Ballot Distribution

- Assume:
 - $E(D; K)$ is encryption of D with K
 - K_{VS} is an ElGamal public key for the voting system
 - Private key k_{VS} is split among all tellers

$E(i,j; K_{VS})$	
	$\text{onion}(i,j)$

- No one knows the map from encrypted LHS to decrypted candidates, i.e. $E(i,j; K_{VS}) \mapsto \langle i,j \rangle$

Ballot Distribution

- Anyone can be permitted to see the ballot, but only voter can learn the LHS decryption
- Distributed reencryption [Zhou et al. '05]
 - Transform $E(D; K_A)$ to $E(D; K_B)$
 - Performed by servers who share k_A
 - Nowhere does D appear as plaintext
 - Voter has $E(i,j; K_{VS})$ reencrypted to $E(i,j; K_V)$
 - But requires $2f+1$ servers/tellers

Ballot Creation

- Goal: Create $E(i,j; K_{VS})$, $\text{onion}(i,j)$
- No single entity can be trusted to create ballots
 - Would learn decrypted candidate map
- Encrypted candidate pair needs to be transformed; we use blinding
 - Ballots created in large sets
 - Each ballot clerk adds a blinding factor and shuffles set
 - By homomorphic property, voter can use distributed reencryption to strip off blinds

Authentication

- Before system can distribute a ballot, must ensure voter is authorized in election
- So voter must authenticate
 - Anything voter knows can be demanded by coercer
 - So like [Juels, Catalano, Jakobsson], we need to enable voter to lie about what he knows
- One idea: Capability is:
$$\text{onion}(S(\text{"valid"}, \text{nonce}; k_{VS}))$$
 - Attach capability to each vote: $\langle vc, 0/1, o \rangle$
 - In final column of mix, capability is stripped to $S(\text{"valid"}, \text{nonce})$
 - Voter can lie by inventing fake capabilities
 - In final column, can detect **anonymized** fakes

Authentication

- Voter can:
 - Give coercer fake capability and let coercer vote
 - Submit any vote (including random) under fake capability
 - Abstain from casting a vote with a fake capability
- Problem:
 - Registrar who creates and distributes capabilities has to be trusted to forget valid capabilities, map from voter to capability issued, etc.
 - Need a distributed onion construction
- Voters must use (sufficiently) anonymous channel to request ballots, submit votes

Conclusions

- Encode ranked ballots in PAV onions
 - (Additive) Condorcet methods
- Eliminated (most of) trusted supervision
 - Ballot creation
 - Ballot distribution
 - (Authentication)

Future Work

- Implementation of CIVS2
 - Jif: What policies can be expressed?
- How can we do anonymous, at-most-once authentication?
 - Distributed onion construction?
- Can ballot distribution failure model be improved using distributed decryption?
- Can we prevent ballot stuffing?