Ballot Secrecy

Essential

- Potential coercion
- Even possibility of disclosure might affect behavior

Hard

- Cell phone cameras
- Leaks to poll workers
- Distinguishing marks

Cryptographic voting has unique problems
Encrypt-on-Cast (e.g. Benaloh, VoteBox)

1. 

2. 

3. 

8/11/09
Voter-initiated Audits

Audit?

Commits

Vote cast

Yes

Verifiably decrypted

Vote doesn’t count

No
Talk Outline

Subliminal Channel Problem

Good News

Bad News

Conclusion
Subliminal Channel Problem
Leaky Bulletin Board [KSW05]

\[ V = E_{pk}(v, r) \]

Want to leak: 011001

\[ E_{pk}(v, r_1) = \ldots 110101 \]
\[ E_{pk}(v, r_2) = \ldots 111001 \]
\[ \vdots \]
\[ E_{pk}(v, r_n) = \ldots 011001 \]

Leak \( t \) bits in expected \( O(2^t) \) work
Only Need to Leak a Few Bits

Don’t need to compromise every voter’s vote

(e.g. 1000 voters)

Reveal how 10% voted with 100 bits

Single out a non-compliant voter with 10 bits
Can Audits Solve This? [GGR09]

Set of $k$ trustees generate all randomness

For each vote,

- Trustees generate: $\pi_1, \ldots, \pi_k$
- $v = E_{pk}(v, r')$ where $r' = f(\pi_1, \ldots, \pi_k)$

If vote audited,

- Machine reveals $r'$ and $\pi_1, \ldots, \pi_k$
- Can verify $\pi_1, \ldots, \pi_k$ with trustees’ public keys

Only for audited votes
Audits Aren’t Enough

Can’t assume a high audit rate — Auditing is cumbersome

Suppose 5% audit
(95% chance of altering 1 ballot without detection)

Steal 1 vote OR Leak 100 bits
10 bits/race with $O(2^{10})$ work, assuming 10 races

Coercion requires corrupting fewer ballots
Good News
Overview

1. Voting machines don’t generate randomness

2. Set of $k$ trustees generate all randomness

3. Anyone can check the randomness on every ballot
El Gamal Encryption

To encrypt,

- Choose random $r$
- $V = (\alpha, \beta) = (g^r, y^r \cdot v)$

(generator $g$, public key $y$)
Before the Election

For each voting machine,

\[
\begin{align*}
\text{Democratic} & : r_{0,0}, g_{r_{0,0}}, r_{0,1}, g_{r_{0,1}}, r_{0,2}, g_{r_{0,2}}, \ldots, r_{0,n}, g_{r_{0,n}} \\
\text{Republican} & : r_{1,0}, g_{r_{1,0}}, r_{1,1}, g_{r_{1,1}}, r_{1,2}, g_{r_{1,2}}, \ldots, r_{1,n}, g_{r_{1,n}} \\
& \vdots \nonumber \\
\text{League of Women Voters} & : r_{k,0}, g_{r_{k,0}}, r_{k,1}, g_{r_{k,1}}, r_{k,2}, g_{r_{k,2}}, \ldots, r_{k,n}, g_{r_{k,n}}
\end{align*}
\]
During the Election

To encrypt vote $v_i$, 

- $\alpha_i = g_{r0,i} \cdot g_{r1,i} \cdot \ldots \cdot g_{rk,i}$
- $\beta_i = y_{r0,i} + r_{1,i} + \ldots + r_{k,i} \cdot v_i$

Combine trustees’ random values

Must use the $i$th values
After the Election

1. For encrypted vote \((\alpha_i, \beta_i)\), check that \(\alpha_i = g_{r_0,i} \cdot g_{r_1,i} \cdot \ldots \cdot g_{r_k,i}\)

2. Rencryption mixnet + decryption

3. To verify \(\beta_i\), check that it decrypts to a valid vote
Why Does This Work?

Corrupted encrypted vote \((\alpha_i, \beta_i')\)

Then, \(\beta_i' = y^{r_{0,i}+r_{1,i}+\ldots+r_{k,i}} \cdot v_i'\)

- If \(v_i'\) is invalid, coercer will be caught
- If \(v_i'\) is valid, it’s equivalent to vote-flipping
Bad News
Vote-flipping Can Leak

\[ V = E_{pk}(v, r) \]

Want to leak: 011001

\[
E_{pk}(v_1, r) = \ldots 110101 \\
E_{pk}(v_2, r) = \ldots 111001 \\
\vdots \\
E_{pk}(v_n, r) = \ldots 011001
\]
Vote-flipping Can Leak (cont.)

Low bandwidth — can fail to leak desired bits

Coercer can deal with this
- Only leak bits in races with enough candidates
- Use an error-correcting code

Previous mitigation strategy won’t work
Conclusion
Conclusion

Subliminal channels are a particular threat to encrypt-on-cast voting systems.

Coercion requires corrupting fewer ballots than vote-stealing (auditing may not catch it).

Verifying the randomness used to encrypt every vote is a partial mitigation.

Vote-flipping itself is a subliminal channel.
On Subliminal Channels in Encrypt-on-Cast Voting Systems

Ariel J. Feldman
Princeton University

Josh Benaloh
Microsoft Research