Lecture 21 - Electronic votes

Boaz Barak

December 6, 2007

> *"Those who cast the votes decide nothing. Those who count the votes decide everything."* (Wrongly?) attributed to Josef Stalin.

Voting mechanisms Paper ballot, absentee mail-in, public/open (raising hands), optical scanning, direct recording electronic (DRE, touchscreen)

Requirements Requirements based on democratic principles: Outcome should reflect the peoples will. **1. Fairness** - one person, one vote **2. Privacy** (required for fairness) More formally we have:

Honest Intentions no vote buying, coercion.

Cast as intended no accidental, malicious miscasting of vote.

Count as cast all votes cast are counted and no more.

Verifiable count independent verification of counts.

Crucial goal: Not just count correctly but make sure that *everybody*, including the losing party, agrees the election was fair. Very hard to do if the election was tight

Attacks Israeli towns with high (over %100?) voter turnout, Italian village attack.

Methods vs. requirements This is summarized in this table (though most entries are subjective and debatable, see Rivest's notes for a different table)

	Paper ballot	Open & public vote	DRE/Touchscreen
Honest Intentions	Y	N	Y
Cast as intended	Y	Y	Y?[1]
Count as cast	?[2]	Y	Y?[3]
Verifiable count	?[4]	Y	N?[5]

This lecture Can we use cryptography to get everything we want?

We will work in a setting where there are k supposedly independent servers/parties that help run the protocol. The security we want is:

[5]Human interface issues, bugs, hacks.

[5]Trust localities to let the right people vote and not let wrong people / impersonator, trust them to count correctly, not stuff ballots.

[5]Aside from bugs/hacks might be even more accurate than hand count. Again, a key issue we largely ignore is deciding who is allowed to vote.

[5]Hard to detect local cheating, count is not universally verifiable (need court order to recount).

[5]No paper trail.

Privacy and receipt freeness Assuming even one of the k servers is honest, every voter's choice is private and in fact even if he wants to, the voter cannot prove to a third party that he made a certain vote.

Universal verifiability Even if all servers are cheating and colluding, if they miscount the votes it can be detected by any party from the broadcasted information.

Mixnets Chaum's suggestion - mix-nets (we'll talk about re-encryption mixnets). Ballot transforms each vote v into $C = Enc(v)$ and the pair (voter id, C) is then broadcasted. There are k server working one after the other, each server takes the list $C_1, \ldots, C_n$ of ciphertexts, re-encrypts each ciphertext and randomly permutes them to obtain a new list $D_1, \ldots, D_n$, and broadcasts the new list and a proof π that that's what he did. At the end they jointly decrypt the final list to get a random permutation of the votes.

Re-encrypting El-Gamal Fix G and g for which DDH holds. Public key is $y = g^d$, private key is d. El-gamal encryption of m is $(g^r, y^r \cdot m)$.

Re-encrypt z, c as $(x^s, c \cdot y^s) = (g^{r+s}, y^{r+s} \cdot m)$.

Verifiable mixes How do you prove that $D_1, \ldots, D_n$ is a mix of $C_1, \ldots, C_n$? (other than just using GMW's theorem that every statement in **NP** has a zero knowledge proof).

First idea: reduce to $n = 2$ Using sorting networks.

Solving the $n = 2$ case Denote $C \sim D$ if they encode the same plaintext. We need to prove that given C_1, C_2, D_1, D_2,

$$\big((C_1 \sim D_1) \wedge (C_2 \sim D_2)\big) \vee \big((C_1 \sim D_2) \wedge (C_2 \sim D_1)\big)$$

or equivalently,

$$\big((C_1 \sim D_1) \vee (C_1 \sim D_2)\big) \wedge \big((C_1 \sim D_1) \vee (C_2 \sim D_1)\big) \wedge \big((C_2 \sim D_2) \vee (C_1 \sim D_2)\big) \wedge \big((C_2 \sim D_2) \vee (C_2 \sim D_1)\big)$$

meaning that it suffices to supply four proofs for a statement of the form

$$(C \sim D) \vee (C' \sim D')$$

Ciphertext equivalence Let's start by showing a proof for $C \sim D$. In other words, prove that $C = g^r, y^r \cdot m$ and and $D = g^s, y^s \cdot m$ or equivalently, if we divide the components, we'll get a pair (z, w) of the form: (g^{r-s}, y^{r-s}). In other words, letting $y = g^a$ and $b = \log_g z$ we need to prove that $w = g^{ab}$. That is, we need a proof that a triple y, z, w is a Diffie-Hellman triple. Chaum and Pederson showed a 3-round public coin proof system for this model that is zero knowledge for honest-but-curious verifier (known as honest-verifier zero knowledge):

Common input: g, G, triple of the form y, z, w where $y = g^a, z = g^b, w = z^{ab}$.

Prover's secret input: a, b

Prover's first message: $(u, v) = (g^s, y^s)$ where $s \leftarrow_{\mathrm{R}} \{0, .., |G| - 1\}$.

Verifier's message: $c \leftarrow_{\mathrm{R}} \{0, .., |G| - 1\}$.

Prover's last message: $t = s + cb$.

Verifier's acceptance criteria: Check that $g^t = uz^c$ and $y^t = vw^c$.

Theorem 1. *The above is a special sound HVZK for the language of DH tuples.*

xor trick It turns out that if we have a 3-round public coins special-sound protocol Π for a language L, then we have such a protocol for the language L' where $(x, y) \in L'$ if $x \in L \ \vee \ y \in L$.

true voter verifiable systems The weak link in the above systems is the machine that encrypts the voter's choice— how do we know that's really what it did? We can try to audit these machines etc.. but we'd really like something better. We'd like each voter to be able to verify that her vote was indeed counted. The challenges in making this happen are that:

1. We don't want to let the voter take home proof that she voted in a certain way.
2. Humans cannot do complicated computations.

Approaches to voted verifiability This idea was initiated by Chaum, and there are two approaches to getting this:

Chaum Systems that give the voter a receipt that she voted in a certain way, but then she gets to take home only half of the receipt (and the other half is destroyed). The verification that the two halves are consistent is simple enough to be done by a human (uses visual cryptography).

Neff, Moran-Naor Systems that use interactive proofs and depend on the order events occur in the voting booth.

Naive suggestions Here are some naive suggestions to get voter-verification:

- The voting machine encrypts the choice v of the voter by choosing randomness r for the encryption algorithm and computing $C = E_e(v; r)$ (where e is the public key). It can give the verifier r, and at home the verifier can check that the value C broadcasted is indeed equal to $E_e(v; r)$.
  The problem with this is that of course r is basically a receipt that the voter can use to prove to a third party that her choice was v.
- The voting machine proves to the voter in zero knowledge that there exists r such that $C = E_e(v; r)$.
  The problem here is that the voter is a human and cannot do the computations involved in verifying zero knowledge proofs.

The solution Suppose that the choice v is either `Alice` or `Bob`. After voter Victoria makes her choice, the voting machine runs *two* zero knowledge proofs: one that C is an encryption of `Alice` and the other is that C is an encryption of `Bob`. Of course one of these is false, but the trick is that for the choice that Victoria didn't make, she will give the machine the challenge in advance, so the machine can run the simulator instead. Victoria gets to take home a printout (possibly signed with the machine's key) that contains her name, the ciphertext C, and the transcripts for the two zero knowledge proofs. She can then verify the transcripts are accepting and that C is indeed the ciphertext broadcasted under her name using her trusted computer (or send it to several watchdog groups that will verify it for her). But even if she voted for Alice, she cannot convince a third party that it was the challenge for the proof that the vote is Alice that she gave the machine later, and the challenge for the proof that the vote is Bob that she gave the machine first.