Secure Elections

should have these properties:

1. Only authorized voters can vote.

2. No one can vote more than once.

3. No one can determine for whom anyone else voted.

4. No one can duplicate anyone else's vote.

5. No one can change anyone else's vote without being discovered.

6. Every voter can make sure that his vote has been counted.

Some voting schemes also require:

7. Everyone knows who voted and who didn't.

Protocol 1.

1. Each voter encrypts his vote with the public key of the Central Tabulating Facility (CTF).

2. Each voter sends his vote to the CTF.

3. The CTF decrypts the votes, tabulates them, and publishes the results.

Problems: The CTF can't tell whether votes come from authorized voters or whether authorized voters vote more than once.

However, no one can change anyone's vote (but why bother?).

Protocol 2.

1. Each voter signs his vote with his private (RSA) key.

2. Each voter encrypts his signed vote with the CTF's public key.

3. Each voter sends his vote to the CTF.

4. CTF decrypts all votes, checks signatures, tabulates the votes, and makes the results public.

This protocol satisfies properties 1 and 2: only authorized voters can vote and no one can vote more than once. Also, no one can change anyone's vote.

The problem is that the CTF knows who voted for whom. You have to trust the CTF completely.

Protocol 3.

1. Each voter prepares ten sets of messages. Each set contains a valid vote for each possible outcome. Each message also contains a randomly generated identification number large enough to guarantee no duplicates.

2. Each voter blinds each of these messages individually and sends them to the CTF.

3. The CTF checks its list to make sure the voter has not submitted his blinded votes previously in this election. It randomly chooses nine of the ten sets of votes and asks the voter to unblind these sets. He does and it checks that these nine sets are properly formed. If they are, then it individually signs each message in the tenth set. It sends them back to the voter and stores his name in the list.

4. The voter unblinds the signed tenth set and is left with a set of all possible votes signed by the CTF. He can tell which is which.

5. The voter chooses one of the possible votes and encrypts it with the CTF's (second) public key. He sends it to the CTF.

6. The CTF decrypts the votes, checks the signatures, checks its database for a duplicate identification number, saves the identification number in the database, and tabulates the votes. It publishes the results of the election, along with every identification number and its associated vote.

The blind signature insures that votes are unique. No one can generate bogus votes or change votes of others because the CTF's private key is secret. A malicious CTF cannot determine how people voted. Each voter can confirm that his vote was tabulated correctly.

However, the CTF could still generate many signed valid votes and submit them itself. If Alice discovers that the CTF has changed her vote, she cannot prove this.

How does the voter "blind" a vote, and how does the CTF "sign" a blinded vote?

Let's say the CTF uses RSA with keys n, e, d.

Let the vote be M.

Voter chooses a random k in $1 < k < n$.

Voter "blinds" (enciphers) the vote by computing $t = Mk^e \bmod n$.

Bank signs t as

$$t^d \equiv (Mk^e)^d \equiv M^d k^{ed} \equiv M^d k \pmod{n}.$$

Voter "unblinds" (deciphers) the signed vote t^d by computing k^{-1} mod n (via extended Euclid) and multiplying:

$$s = t^d k^{-1} \bmod n \equiv (Mk^e)^d k^{-1} \equiv$$

$$\equiv M^d k^{ed-1} \equiv M^d \pmod{n}.$$

To "open" (decipher) the 9 t, voter tells the CTF M and k for each. The CTF verifies that each $t = Mk^e$ mod n.

The next protocol has the six properties listed above and these two:

7. A voter can change his mind (delete his old vote and cast a new one) within a certain time period.

8. If a voter finds that his vote is miscounted, he can identify and correct the problem without jeopardizing the secrecy of his ballot.

Protocol 4.

1. The CTF publishes a list of all eligible voters.

2. Before a deadline, each voter tells the CTF whether he intends to vote.

3. At this deadline, the CTF publishes a list of all eligible voters participating in this election.

4. Each voter receives a unique identification number I, e.g., by mental poker or blind signature protocol.

5. Each voter generates a public/private key pair k, d. If his vote is v, he sends the message $I, E_k(I, v)$ anonymously to the CTF.

6. The CTF acknowledges receipt of the vote by publishing $E_k(I, v)$.

7. Each voter sends the message I, d to the CTF.

8. The CTF decrypts the votes. At the end of the election it publishes the results of the election and, for each different vote, the list of all $E_k(I, v)$ values which contained that vote.

9. If a voter sees that his vote is not properly counted, he protests by sending $I, E_k(I, v), d$ to the CTF.

10. If a voter wants to change his vote from v to v', he sends $I, E_k(I, v'), d$ to the CTF.

Steps 1 through 3 of Protocol 4 are preliminary to the actual voting. They reduce the ability of the CTF to add fraudulent votes.

If two voters get the same I in Step 4, the CTF discovers this in Step 5. It creates a new identification number I', chooses one of the two votes, and publishes $I', E_k(I, v)$.

The owner of that vote recognizes it and votes again by repeating Step 5 with the new I'.

In Step 6, each voter can check that his vote is counted accurately. If it is miscounted, he can prove this in Step 9.

One limited problem is that the CTF could make up fraudulent votes for people who respond in Step 2 but don't actually vote.

A more serious problem is that CTF could neglect to count a vote. Alice could claim that the CTF deliberately neglected her vote, while the CTF could claim that she never voted.