6.857 Computer and Network Security **Fall Term, 1997**

Lecture 22 : November 20, 1997

Lecturer: Ron Rivest *Scribe: Sabra-Anne Kelin*

1 Handouts

- Problem Set 9
- *A Practical Voting Scheme for Large Scale Elections*, by Fujioka, Okamoto, and Ohta (1992)

2 Topics

- Electronic Voting

We want to design an electronic voting system. Our system should be able to handle a large, distributed voting population, such as students in a university or citizens in a nation.

2.1 System Requirements/Desiderata

1. Each person can vote at most once.
2. Votes are anonymous.
3. The count should be verifiable by a reliable source. Also, each voter should be able to see that her vote was counted correctly.
4. No correlations are evident.
5. Only authorized voters may vote. (Should a public voter list exist?)
6. Each voter is authenticated to ensure that she is casting her own vote.
7. Some mechanism exists to handle failures and discrepancies (see 3).

8. No receipts are given to the voter which indicate how the voter voted. This prevents voters from selling their votes and prevents crooked politicians from coercing voters to vote in a particular way.
9. A deadline exists to mark the end of the election.
10. There is a way to write-in candidates.
11. The system authenticates itself to the voter. Otherwise, people can set up phony voting terminals/programs.
12. System is efficient, easy to implement, verifiable, easy-to-use, and scalable.
13. Should you be able to change your vote once it has been cast?
14. There is a method by which you can appoint someone as your proxy.
15. System is continuously available and robust.

2.2 Techniques Used in *A Practical Voting Scheme for Large Scale Elections*

Commitments Commitments ensure that voters cannot change their votes after they have been cast. After a voter has committed to her vote and has sent in her commitment, she sends in the key that unlocks the commitment and reveals the vote.

Blind Signatures Blind signatures allow someone to sign a message without knowing what the message says. In this scheme, the administrator signs all the votes to indicate that they were cast by legitimate voters.

The idea behind blind signatures is to randomize the message before the administrator sees it.

For example, signing a message using RSA looks like

$$m \rightarrow m^d$$

where m is the message and d is $1/e$. (e is the encryption key)

To implement blind signatures, the voter multiplies her message by r^e, where r is a random number. She then submits this message to the administrator. After the message has been signed blindly, the administrator sends the message back to the voter, who derandomizes it. (m is the voter's commitment)

$$m \rightarrow r^e m \rightarrow r m^d \rightarrow m^d$$

Anonymous Channels In this system, a central server acts as an anonymous channel. The server delays messages, reorders them, and strips off identifying information. This way, messages coming out of the server cannot be traced back to their sources. Notice that the server is trusted. Another example of an anonymous channel is an anonymous email server.

2.3 Actual Design

Parties Involved

- Administrator/Registrar – Possesses a list of legal voters. Signs vote blindly if vote was cast by legal voter.
- Voters
- Anonymous Channel (AC)
- Counter – Machine that tallies the votes.

Voting Procedure

The voting procedure is divided into two phases. The first phase deals with people casting votes. The second phase, which starts when the voting deadline has passed, deals with tallying the votes and checking their validity.

Phase 1:

1. Voter obtains ballot (by logging into machine, etc.)
2. Voter indicates her choices on the ballot.
3. Voter commits to her choices.
4. Voter blinds the commitment by randomizing it.
5. Voter signs the blinded commitment and sends it to administrator.
6. Administrator checks that the commitment was signed by a registered voter who hasn't yet voted in this election. Administrator signs the commitment and sends it back to the voter.

7. Voter unblinds vote by derandomizing it.

8. Voter sends resulting message (a signed commitment) to the Counter via the AC.

*** Deadline for submitting votes ***

Phase 2:

1. Counter publishes all votes received in the form {commitment, signed commitment}. (Counter received only signed commitment, but since the Counter is part of the voting system, it knows the algorithm used by the Administrator to sign the commitment. Thus it can uncover the original commitment.) Voter checks that her vote is listed, since she knows her original commitment and the signed commitment she sent to the Counter.

2. Voter sends decryption string (to open commitment) to the Counter via the AC.

3. Counter opens up all commitments and lists the votes in the form {commitment, signed commitment, vote, decryption string}. Now people can check that for each vote, opening the commitment using the decryption string yields the vote in question.

2.4 Interesting Characteristics of the Design

- Votes can be sold.
- Administrator can make up votes. (Administrator must be trusted implicitly.)
- Counter cannot make up votes.
- Voters cannot cheat.
- What if the voter doesn't send in her decommit string? Solution: impose two deadlines. One to send in the commitment, one to send in the decommit string.