Virtual Election Booth
CS419—Spring 2013

Out: March 1st. Due dates: (in phases)

This project requires you to implement the secure election protocol described in Schneier's book on Applied Cryptography, p. 127 (Voting with Two Central Facilities).[1] The implementation will provide a secure way for people to vote online, which eliminates the hassle of physically being present at designated election locations.

Since computerized voting will not replace general elections unless there is a protocol that both maintains individual privacy and prevents cheating, the ideal protocol must meet these requirements:

- Only authorized voters can vote.
- No one can vote more than once.
- No one can determine for whom anyone else voted.
- No one can duplicate anyone else's votes.
- Every voter can make sure that his vote has been taken into account in the final tabulation.
- Everyone knows who voted and who didn't

Your design should use two central facilities: Central Tabulating Facility (CTF) and Central Legitimization Agency (CLA). CLA's main function is to certify the voters. Each voter will send a message to the CLA asking for a validation number, and CLA will return a random validation number. The CLA retains a list of validation numbers as well as a list of validation numbers' recipients to prevent a voter from voting twice. Then, the CLA completes its task by sending the list of validation number to the CTF. CTF's main function is to count votes. CTF checks the validation number against the list received from the CLA. If the validation number is there, the CTF crosses it off (to prevent someone from voting twice). The CTF adds the identification number to the list of people who voted for a particular candidate and adds one to the tally. After all the votes have been received, the CTF publishes the outcome.

Protocol

The following excerpt from Schneier's book, p. 127 describes the secure voting protocol. It is recommended that you read the chapter (available via the "Resources" section in Sakai) for more discussion and background.

The following protocol uses a Central Legitimization Agency (CLA) to vertify voters and a seperate CTF to count votes.

1. Each voter sends a message to the CLA asking for a validation number.
2. The CLA sends the voter back a random validation number. The CLA maintains a list of validation numbers. The CLA also keeps a list of the validation number's recipients, in case someone tries to vote twice.
3. The CLA sends the list of validation numbers to the CTF.

[1] Bruce Schneier, Applied Cryptography, 2nd Edition, John Wiley and Sons, 1996, ISBN 0-471-11709-9

4. Each voter chooses a random identification number. He creates a message with that number, the validation number he received from the CLA, and his vote. He sends this number to the CTF.

5. The CTF checks the validation number against the list it received from the CLA in step 3. If the validation number is there, the CTF crosses it off (to prevent someone from voting twice). The CTF adds the identification number to the list of people who voted for a particular candidate and adds one to the tally.

6. After all votes have been received, the CTF publishes the outcome, as well as the lists of identification numbers and for whom their owners voted.

The system should be implemented using your choice of C, C++, or Java. Your implementation will consist of 3 independent programs that communicate over the network. Two of these programs implement the CLA and CTF. The third is the user interface. This can be implemented either as a traditional GUI program, or as a web server serving HTML pages. **All communications between these programs should be secured using the secure sockets layer (SSL).** You must use the SSL libraries available for your choice of implementation platform (C, C++, Java) to enable secure communication. **Do not attempt to roll your own encryption library.**

Timeframe and Deliverables

The project will proceed with several checkpoints:

1. Project design document: due on March 30, 2013. This document must identify your team members (you can work in teams of size two) and must contain three sections. The first section must present the steps of the protocol in detail. The protocol should not be expressed in words, but rather as a sequence of steps, in much the same way that we discussed protocols in class. If you can include a picture in this section describing the steps of the protocol, that will be great. The second section must present the system design: i.e., how you plan to implement each of the components of the protocol. The third section must identify the security requirements that the protocol achieves, and prove how the protocol achieves these requirements. These are the same requirements identified in the second paragraph above.

2. Project demos: May 2, 2013. You are expected to have a working demo of your project by this time, and must demonstrate your projects to the TA and me. As this date approaches, we will schedule demo slots for which you will have to sign up. Expect to demonstrate a working program, be familiar with the code and be prepared to answer questions about it.

3. Project submission: May 6, 2013. The final project report and the code are due on the last day of classes. I will create an entry on Sakai for you to upload a tarball of the code and a PDF of the final project report. This report must enhance the design document with implementation notes and performance measurements of your protocol.

Credits

This project is based upon a similar project offered in the course "Introduction to Information Security" at the University of Wisconsin-Madison.