Secure E-voting System

Rohit Koul   Aditya Thakur

Abstract

We present a system for secure electronic voting which maintains individual privacy and prevents tampering and cheating by voters. By using two central facilities it further ensures that no single entity can fix the election. We present the rationale behind the protocol used and an overview of the system.

1 Introduction

Electronic voting systems provide significant advantages over conventional voting and save significant resources in the process of setup, distributing, voting, collecting and counting of ballots. However, for all the above savings, electronic voting systems must provide security in each stage of the process to avoid any compromise on the authenticity of the results. These concerns of verifiability and security need to be addressed appropriately especially if electronic voting is performed over the internet. This project aims at providing one such secure framework by implementing a secure election voting protocol based on [2] and [1] that maintains privacy and ensures fairness of the elections.

This design document describe the various entities, design model and message flow of our electronic voting system and will facilitate the implementation of our project.

The document is organized as follows: §2 lists the requirements of the system, §3 describes the design of the voting system, the entities involved in the system and the underlying protocol, §4 describes the system architecture and implementation details of the protocol used, §5 gives a description of the threat model, and how the protocol meets the prescribed requirements and §6 lays out the limitations of the system.

2 Design Requirements

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

3 Protocol Overview

3.1 Protocol Entities

Our protocol involves the following three entities:

- *Voter.* The person who will be using the secure voting system to cast his vote.

- *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 Central Tabulating Facility(CTF).
- *Central Tabulating Facility (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 already present, the CTF crosses it of f(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.

3.2 Protocol Stages

The protocol consists of the following four stages:

1. The voters communicate with the CLA and get their validation numbers
2. CLA and CTF communicate wherein CLA sends the list of valid validation numbers to the CTF
3. The voters send their vote to the CTF
4. CTF publishes the result

It is assumed that the stages are sequential, i.e., stage 2 occurs only when stage 1 is over, stage 3 follows stage 2 and so on. This prevents certain asynchronous events from happening (such as a voter trying to cast his vote to the CTF even before CLA got a chance to send the list of validation numbers to it). In the wake of how real life voting happens, wherein the dates/period for registration and voting are announced separately, we believe this is a reasonable assumption to make. We further discuss the rationale of this assumption in §5.

3.3 Message Flow

At a high level, the flow of messages over the communication channels is as follows:

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.
4. Each voter chooses a random identification number. She creates a message with that number, the validation number he received from the CLA, and her vote. She 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 of (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.

3.4 Underlying security support

To support the necessary security requirements of our protocol we will be using SSL (Secure Sockets Layer) in the underlying implementation. SSL is a cryptographic protocol that provides secure communication using message encryption and authentication between two communication parties. This helps us in implementing the privacy and authentication needs of our protocol.

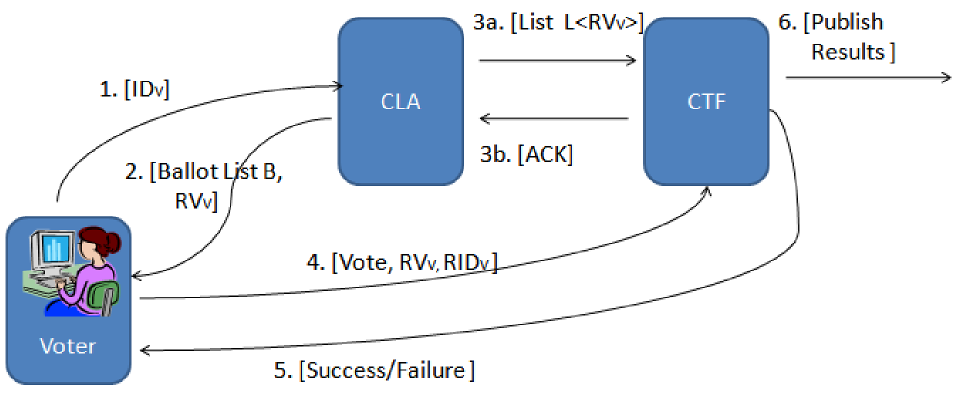

Figure 1: Message Flow

4 Implementation Overview

4.1 Identity verification

All the three entities involved in the process of electronic voting need to authenticate themselves. This is required so that fake entities do not obtain validation numbers from the CLA. Also, we need to ensure that no fake or fraud entity poses as CLA or CTF and compromises the security of the electronic voting process. This necessitates the requirement of public key certificates (Class 2) from the voters, CLA and CTF from a trusted certificate authority (CA) such as VeriSign.

Class 2 certificates require that the individual should present unique identification like Social Security Number (SSN) to the CA to make sure that the public key certificates are bound to the appropriate person, agency (CLA/CTF). In this protocol implementation, all entities pre-obtain the public certificates for appropriate validation during SSL handshake.

4.2 Other Constraints

The electronic voting system is implemented as different processes, operating and communicating from different computing terminals. These entities exist with the following constraints:

1. Only one CLA exists at any point of time. This CLA will securely communicate with different voters and CTF.
2. Only one CTF exists at any point of time. This CTF will securely communicate with CLA and voters.
3. Multiple voters may exist at any given point of time. However, each of these voters should have unique Class 2 public key certificates. These voters communicate securely with the CTF and the CLA.

As discussed earlier in §3.2, the voting registration period and the voting cast period is publicly known and hence the voters, CLA and CTF are aware of the time period remaining until which voters could cast votes and CLA/CTF will accept them. The CLA does not register any voter after the registration period is over. Similarly CTF does not count any vote after the voting period is over.

4.3 Message Flow Implementation

This section describes the communication between different entities. The message flow protocol that we will implement in our electronic voting system is represented below. It is based on an underlying SSL protocol that ensures authentication and privacy of this communication. We first describe the terminologies that are involved in the figure.

RVv- Random Validation number for Voter V generated by CLA.
IDv- Unique Identifier for Voter V such as Social Security Number.
RIDv- Random Identification number generated by Voter V.
VOTE- The vote casted by Voter V.

Message Exchanges between Voter and CLA

1. Voter to CLA

   The voter requires a validation number to begin voting and CLA is responsible for generating them. The voter securely sends an identification number [1] to the CLA and requests for a Validation Number.
   Voter $V \rightarrow$ CLA : IDv

2. CLA to Voter The CLA generates a validation number and sends it to the voter along with the list of candidates.
   CLA $\rightarrow$ Voter V: Ballot List B || RVv

Message Exchanges between CLA and CTF

This involves the exchange of information between the CLA and the CTF. The CLS has to send the list of validation numbers it generated. Communication happens over SSL with client authentication enabled to ensure both CLA and CTF authenticate to each other. The CLA doesn't send the identification number that the voter had specified to generation of the validation number (in Steps 1, 2).

CTF also send an acknowledgement to CLA that it received the list. If the ACK gets lost or delayed and/or is not received before the CLAs application level timer for the same expires, CTF re-transmits the list.

3. (a) CLA to CTF
   CLA $\rightarrow$ CTF : List$<$RVv$>$: - list of voter validation numbers

3. (b) CTF to CLA
   CTF $\rightarrow$ CLA Ack

Message Exchanges between Voter and CTF

The voter casts her vote. The vote needs to be securely communicated to the CTF which is responsible for counting and publishing results. The important thing to note here is the voter's identity shouldn't be known to the CTF. The CTF should receive the voter's validation number, a random identification number and the vote. Once the CTF receives this information, it would cross reference the validation number with the list it received from the CLA and updates the list for all voters who have voted. If the vote is valid (i.e. contains a 'good' validation number, hasn't voted already, and is within the electoral period), the vote is accepted and a message is sent back to the voter indicating this. If anything goes wrong in this process, a failure message is sent.

4. Voter to CTF
   Voter $\rightarrow$ CTF : VOTE || RVv || RIDv (Vote, Validation Number and Random Identification Number).

5. CTF to Voter
   CTF $\rightarrow$ Voter V: Success/Failure (Acknowledgement of whether the vote was successful or failed due to any reason.)

[1] IDv is not absolutely necessary since the voter's identity is already bound to the certificate presented during SSL negotiation. In our implementation we check whether the IDv presented in this message is the same as that used during SSL negotiation.

Publishing of Election Results

Once all voters have cast their vote, the CTF publishes the results (step 6). The result also includes a list of random identification numbers of voters who cast their vote to a particular candidate. This allows a voter to ensure that his/her vote has been included in the counting phase. Furthermore, the result also includes the list of the validation numbers of the voters who voted. This allows the CLA to verify that only valid voters' votes were counted viz. if the number of validation numbers published by the CTF is not equal to the number of votes cast, then there is something wrong. Also, if the published list contains validation numbers which were not issued by the CLA, then the CLA can detect this.

5 Protocol Rationale and Threat Model

This section describes the threat scenarios and how our model deals with them in order to meet all the requirements listed in §2.

Requirement 1: Only authorized voters can vote

Threat 1: Can a non-authorized voter sneak in?
Counter: CLA grants a validation number to valid voters only. The voter and the CLA communicate via SSL. The voter identity is authenticated via a Class 2 digital certificate during the communication. The validation number is sent over the SSL connection and hence encrypted. The use of SSL prevents any man-in-the-middle attack since the voter and CLA authenticate each other. An adversary can attempt to guess the validation number but the validation numbers can be drawn randomly from a range much larger than the number of voters.

Threat 2 - Malicious CLA: Spurious votes can be put in by a malicious CLA. The CLA can provide validation numbers to unauthorized votes.
Counter: This can be mitigated to some extent by making the CLA publish a list of valid voter IDs. The CTF can check if it receives any more votes than the number of voters on the valid voter list.

Threat 3 - Malicious CTF: If not all valid voters cast their vote, the CTF can make spurious votes in place of the actual voters by generating a random identification number and the vote.
Counter: There can be some measures to work around this. The CTF sends a list of validation numbers to the CLA for which votes were received. (done over SSL). The CLA then publishes a list of voter IDs for which the votes were received. The validation numbers are not made public. Voters who did not cast can check if their IDs occur in the above list. If yes, then one could be sure that either the CTF and/or the CLA might be compromised.

Requirement 2: No one can vote more than once

Threat: A Voter can try to vote more than once.
Counter: The CLA grants a validation number exactly once to an authorized voter, even if the voter tries to obtain a validation number multiple times. Hence, an authorized voter receives exactly one validation number; which the CTF ensures can be used exactly once. The CTF crosses out the validation number the first time a vote using the validation number it received and hence prevents re-voting. This also makes a replay attack unsuccessful, i.e., if a malicious person tries to simply replay the vote message from an authorized voter to the CTF, the CTF will be able to ignore this replayed message.

Note: It is worth mentioning that one could argue that in certain parts of the world, it might be perfectly legal to change ones vote as many times as possible within the designated voting period. We believe it is a reasonable argument and the requirement put forth in our original protocol is somewhat more strict in this sense. A less strict requirement and the one we actually care about should be "No voter is able to cast a countable vote for more than one candidate". Hence, even if a voter is allowed to vote multiple times, only the last cast should be taken as valid.

This requirement is easy to enforce since the CTF could verify if the voter has voted before and simply decrement the tally for the previously voted candidate and increase the tally for the new candidate being voted for. Also the random identification numbers generated by the voters should be random enough and drawn from a wide range so that someone else is not able to recast on someone elses behalf, failing which the stricter requirement is recommended.

Requirement 3: No one can determine for whom anyone else voted

Threat: The voter's vote maybe known to others apart from CTF.
Counter: The voter remains anonymous from the CTF. Only the CTF is authenticated when a vote is cast during the SSL connection. The CTF just sees the validation number and hence cannot associate this number to the voters identification.

The published list by the CTF contains random identification numbers sent in by the voters which the CLA cannot map this random ID to the voter itself. Again, collusion between the CLA and the CTF can undermine voter anonymity. Since all communication is done via SSL, an eavesdropper cannot get the voter, validation number mapping sent by the CLA or the vote sent to the CTF.

Requirement 4: No one can duplicate anyone else's votes

Threat: A voter's vote maybe duplicated.
Counter: Duplication doesn't work because the CTF maintains a list of validation numbers for which votes have been received. Again as discussed previously, a malicious CTF can generate a vote for some voter who did not cast her vote. But such a CTF can be detected.

Requirement 5: Every voter can make sure that her vote has been taken into account in the final tabulation

Threat: A voter's vote may be omitted in the final tabulation.
Counter: The CTF publishes a list of identification numbers coupled with the vote. Voters can check this list to ensure their vote was cast correctly and accounted for.

Requirement 6: Everyone knows who voted and who didn't

Threat: The status of whether a vote has been cast by a voter is not clear. (It is not clear who voted and who did not).
Counter: The CTF sends a list of validation numbers to the CLA for which votes were received (done over SSL). The CLA then publishes a list of voter IDs with information that the vote for this ID was received or not.

Importance of staged voting process

As discussed in §3.2, we argue that the voting process should happen in different sequential stages/phases. Let us consider a case wherein this is not such a strict requirement and asynchronous voting events are allowed by the system. In such a scenario, there are no separate registration (phase 1) and voting (phase 3) periods, instead voters could register and vote throughout the electoral duration. The steps 1, 2 and 3 of the message flow will be as follows:

1. Each voter sends a message to the CLA asking for a validation number.
2. (a) After performing necessary validation checks, CLA generates the validation number for the voter and sends it to the CTF.
   (b) The CLA also sends the voter back the random validation number. It maintains a list of such validation numbers and their recipients, in case someone tries to vote twice.
3. While steps 1 and 2 are happening, a previously authenticated voter chooses a random identification number. She creates a message with that number, the validation number he received from the CLA, and her vote and sends this number to the CTF.

4. The CTF checks the validation number against the list it received from the CLA in step 2.a for every registered voter. If the validation number is there, the CTF crosses it of (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.

5. 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.

As is clear from the steps above, there could be scenario wherein the CTF does not received the validation number from the CLA (generated in step 2a) due to network failure/delays before the said voter tries to cast her vote to the CTF. One possible solution is to send back an error message asking the voter to try later. However, an adversary could quickly abuse this system to cause a DDOS attack on CTF. Besides, it is too much of a hassle for the voter.

Another approach is to buffer such votes for a short period of time and wait for the validation number from CLA to arrive in that period. However, it is now even simpler for an adversary to cause the CTF to slow down or crash by taking over its buffer space by a flood of invalid validation identifiers (akin to SYN-FLOOD attack in TCP). It is also not clear, what the CTF should do if the validation number from CLA were not in arrive in that short period of time. If the CTF chooses to discard such a vote, it could be discarding a valid vote.

A third approach is also possible in which explicit synchronization between the CTF and CLA between steps 2a and 2b, viz. CLA sends a "voter list update" to the CTF and the CTF acknowledges this update. But this message could also be lost or delayed, in which case we are back to the original problem.

Thus, of all the possible solutions, the staged voting process we describe is the simplest and less prone to network error.

6 Limitations

- Though a malicious CTF can be detected, the CLA is still a trusted authority in some respects. It can certify ineligible voters and also certify eligible voters multiple times. This risk could be minimized by having the CLA publish a list of certified voters (but not their validation numbers). If the number of voters on this list is less than the numbers of votes tabulated, then something is awry. However, if more voters were certified than votes tabulated, it probably means that some certified people didn't bother voting.

- If the CLA and CTF collude, then they can figure out who voted for whom. This is a basic assumption behind the design of the protocol.

- CLA and CTF are susceptible to DOS and DDOS attacks. Appropriate measures should be taken to prevent against this.

- In our implementation of the protocol, no information is stored on disk. Thus, if teh CLA or CTF processses crash, then all the state is effectively lost. To prevent against this the data should be encrypted and stored on disk regularly in a manner which is amenable to recovery.

- In the current implementation, the ballot list is simply stored on disk in an unencrypted form and is read by the CLA and CTF. Any modification of this ballot list, e.g. removing a candidate name, could compromise the election result.

A DTDs

This section describes the various DTDs used for the messages described in detail in §4.3.

Message Exchanges between Voter and CLA

1. Voter to CLA

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT MSG (IDENTIFICATION)>
<!ELEMENT IDENTIFICATION (#PCDATA)>
```

2. CLA to Voter

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT MSG (VOTER_VALIDATION_NO, TIME_LIMIT, BALLOT_LIST)>
<!ELEMENT VOTER_VALIDATION_NO (#PCDATA)>
<!ELEMENT TIME_LIMIT (#PCDATA)>
<!ELEMENT BALLOT_LIST (CANDIDATE_NAME+)>
<!ELEMENT CANDIDATE_NAME (#PCDATA)>
```

Message Exchanges between CLA and CTF

3. (a) CLA to CTF

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT MSG (VALIDATION_NO_LIST)>
<!ELEMENT VALIDATION_NO_LIST (VOTER_VALIDATION_NO+)>
<!ELEMENT VOTER_VALIDATION_NO (#PCDATA)>
```

3. (b) CTF to CLA

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT MSG (ACK)>
<!ELEMENT ACK (#PCDATA)>
```

Message Exchanges between Voter and CTF

4. Voter to CTF

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT MSG (VOTER_VALIDATION_NO, RANDOM_IDENTIFICATION_NO,VOTE)>
<!ELEMENT VOTER_VALIDATION_NO (#PCDATA)>
<!ELEMENT RANDOM_IDENTIFICATION_NO (#PCDATA)>
<!ELEMENT VOTE (#PCDATA)>
```

5. CTF to Voter

```
<?xml version="1.0" encoding="UTF-8"?>
<!ELEMENT MSG (SUCCESS)>
<!ELEMENT SUCCESS (#PCDATA)>
```

References

[1] Arto Salomaa. *Public-key cryptography*. Springer-Verlag New York, Inc., New York, NY, USA, 1990.

[2] Bruce Schneier. *Applied cryptography (2nd ed.): protocols, algorithms, and source code in C*. John Wiley & Sons, Inc., New York, NY, USA, 1995.