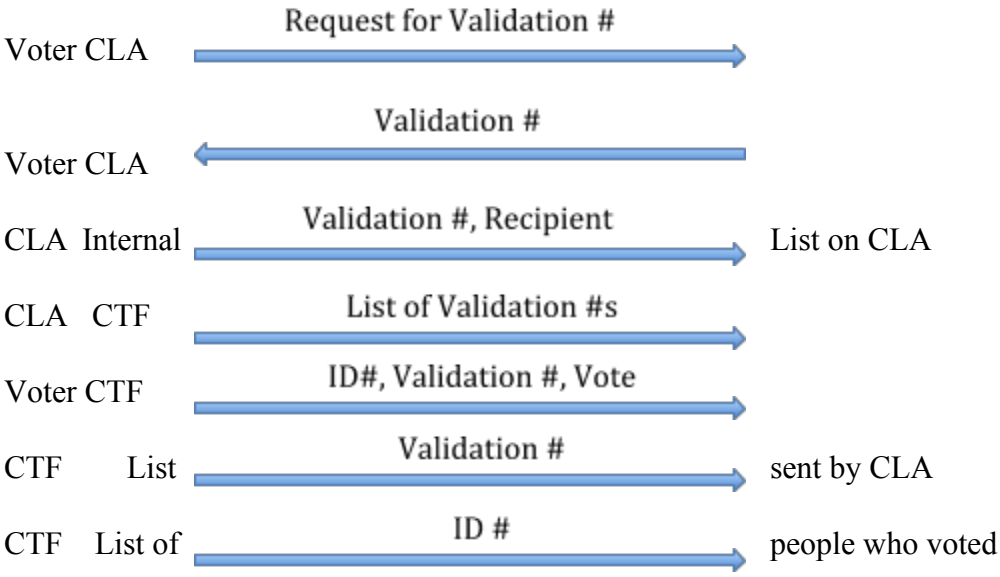
Computer Security Project Design Document
Virtual Election Booth
By: John Cierpial and Edward Zaneski

Section One



Section Two

We propose that our protocol for the CLA and CTF will be implemented using the Go programming language. The Go programming language allows for concurrency, and is one of the key features of the language. It is our belief that by utilizing concurrency, we can maintain not only a secure protocol, but a fast and efficient one as well. The front end will be designed as a webserver serving HTML pages. The HTML pages will then transmit its information onto the CLA when the voter requests a validation number, which will be running in the background retrieving the data from the webserver and transmitting information back to it so that the voter can receive its validation number. The CLA will also maintain an internal data structure containing a list of all validation numbers given as well as who the recipient of that validation number was. This cycle will continue until "voter registration" closes, at which point the CLA will finalize its storage of data, and pass the list of validation numbers given onto the CTF. Since the CTF portion will also be programmed in Go, data sharing should be a trivial matter.

The CTF will then create its own internal data structure where it will store a list of validation numbers given to it by the CLA. There will then be a second HTML page on the webserver where the user can input their random identification number, validation number, and their vote. This information, when submitted, will be passed from the webserver to the CTF, where the validation number will be compared to the list of existing validation numbers, removed from the list if that number is valid, and then the CTF will store in a second data structure the identification number, validation number, as well as vote entered by the voter as a triple. The CTF will also increment the vote counter for the number of votes the candidate has received. This process will continue for each voter until the voting process has ended. By that point, the CTF will generate a third data structure containing as a tuple the identification number as well as who that number voted for. It will, in addition, publish the outcomes of the election. All of this publishing of information will again be trivial as the system has been keeping track of this information throughout the voting process, and thus, will not need to aggregate all of the information at the end. This will improve the overall efficiency and runtime of the CTF process.

Section Three

Since this is an election voting system and all votes should remain confidential, safety and security of data is most important. This protocol first maintains the concept of anonymity of votes. By publishing an identification number along with the vote associated with that number, no one except for the voter knows who anyone voted for. This allows for voters to confirm their votes were tallied correctly while at the same time ensuring no one else knows whom they voted for. Because the CLA is the only process that knows which validation number is associated with a particular name, even the vote day process (CTF) does not know what person cast a particular vote. They simply identify the person by some random number.

This protocol also ensures that no one can vote more than once. The CLA system ensures each person can receive only one validation number while the CTF ensures that validation number is used to vote only once.

Additionally, this protocol ensures that no one can duplicate anyone else's votes due to the use of the secure sockets layer (SSL). SSL ensures that data is transmitted between the sender and receiver in a secure manner and is not subject to interception. This idea of wrapping all communications in SSL ensures that the validation number needed for someone to vote is transmitted only to them by the CLA and cannot be used by anyone else, unless the end user somehow divulges their validation number (this is not a flaw of the protocol, but rather a flaw in human psychology).

Each voter will be able to ensure that their vote has been taken into account in the final tabulation by consulting the list published by the CTF at the end of the election where it lists the identification numbers and who that number voted for. As long as the voter remembers their random identification number, they can ensure their vote was correctly accounted for.

While this system is not the ideal protocol because it does not meet the requirement of allowing only authorized users to vote (the system allows anyone to request and receive a validation number from the CLA) and because it does not allow everyone to know who voted and who didn't (the CTF only has validation numbers, identification numbers, and recorded votes. The CLA has a list of names, but never transmits that information to the CTF). However, this proposed protocol is quite secure and effectively ensures that not only votes are properly authenticated, but that voters remain anonymous and their vote remains known only to themselves.

Section Four

In terms of implementation, we elected to use Node.js and JSON to create both our CLA and CTF servers. We chose this implementation as many of the desired features we sought in our CTA and CTF servers such as SSL, were easy to implement using Node.js. Not only was the implementation simplified, but the code is relatively straightforward, easy to read, and not very long either. This choice also allowed for easy error checking as well as identity verification. For example, one of our checks is to ensure that the CLA is signing its messages to the CTF with the correct key. If the key is invalid or somehow manipulated, the CTF is able to detect this

and will actually return an error message and prevent the user from registering to vote. Because security is so important in an election process, there was no aspect that was left untouched. An example of this is our decision to check for the possibility of an XSRF attack. Should the session key somehow be manipulated on the client side, the server will detect the invalid key when information is sent to it, and will reject either the voter registration or ballot-voting attempt.

When writing this implementation, we found an interesting "feature/bug" in that the browser would only accept the certificate of the first server that runs, whether it be CLA or CTF. This is due to the fact that both the CLA and CTF are running on the same host (in our case, localhost). Because the browser only accepts one certificate from the host, both the CLA and CTF are forced to use the same SSL key. In the real world, however, the CLA and CTF would be running on two different servers with different hostnames, and thus the certificates would be accepted by the browser. To get around this issue for the purpose of our demonstration, we had both the CLA and CTF use the same key.

Regarding performance, not only is our implementation very stable, it is scalable as well. The stability and scalability of this implementation lends itself to the overall simplicity when designing and building both servers. The servers can handle multiple clients and is limited only by the computing power of the machine it is hosted on. Because the code is lightweight and information is exchanged between the CLA and CTF in real time, there is no large processing requirement at any point in the election booth execution, nor is there a need to store large amounts of information besides what is necessary for the CLA and CTF servers to function.