Secure Virtual Election Booth with Two Central Facilities

Janga Sireesha (*sj2@cec.wustl.edu*)
So-In Chakchai (*cs5@cec.wustl.edu*)
Department of Computer Science Washington University in St. Louis, USA

December 14th 2005

Abstract

This project implements a modified secure election protocol with two central facilities, CLA and CTF, which accomplishes the objectives of a secure election. With the combination of public/symmetric keys and hashing function, this two server scheme overcomes the drawbacks of the voting blind signatures and voting with a Single Central Facility in both privacy and cheating. This modified protocol could be adapted to implement in practice; however, there is a security tradeoff for some functionalities; for example, recording who voted or whom voters voted for.

1. Introduction

With the vivid growth of the internet and World Wide Web, it would be more convenient and much more secure to have the online voting instead of the conventional election in order to get rid of the hassle of physically being present at designated election locations. However, this online voting scheme can not be used if we can not maintain the individual privacy and prevent cheating. This can be accomplished with the help of today's amazing technology of computer networks and cryptographic techniques.

Generally it is assumed that voting over the internet is similar to every day's commercial transactions. But there is more to consider for implementing voting online. These days a lot of scams and security violations have been heard about financial transactions. Thus, we have to consider a lot of security issues while implementing the voting online. A secure and dependable election is the most important key for a democratic country to successfully flourish. If we screw up the elections by implementing insecure policies, it is more hazardous to the integrity of the whole administration of the country itself and it looses people's faith in the government. Therefore a lot of caution should be exercised when implementing the online voting. This implies that the voting requires a higher level of security compared to the other financial transactions conducted over the internet. There exists quite a bit challenge inherent to implement the online voting.

Another more important issue of the election, the right to vote, is not transferable and must not be delegated, sold, traded or given away. This type of requirement is fundamentally different from the normal transactions that can be done on the internet. Also the secrecy of the voting should be maintained and nobody but the voter knows whom he voted.

With all these complicated issues to consider on hand, there are several voting protocols lying out there trying to address these inherent problems in the structure and requirements of the online voting. They tried to alleviate some of the problems by making use of existing cryptographic techniques and today's networking technology.

Our project includes the extensive research of the existing protocols, identifying their security flaws and implementation deficiencies. We also summarize the essential requirements of a secure voting protocol before diving into the actual protocols. We give a brief description of the existing protocols and narrate some of the advantages and disadvantages of each of those protocols. Finally, we give the slightly modified version of the existing protocols which alleviates the security drawbacks and confirms to the given security requirements.

1.1 Secure Voting Requirements

According to [2], a secure protocol for the online voting should be implemented in order to maintain both privacy and prevent cheating; the requirements to achieve these objectives are as follows;

- 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.
- No one can change anyone else's vote without being discovered.
- 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. (Optional)

1.2 Voting Protocols

1.2.1 Simplistic Voting Protocol # 1

- Each voter encrypts his vote with the public key of a Central Tabulating Facility (CTF).
- Each voter sends his vote to the CTF.
- The CTF decrypts the votes, tabulate them and makes the results public.

This is a very simple protocol and is having a lot of security flaws. Of course this obeys one of the voting requirements that no one can change any one else's votes. But it authorizes a voter multiple times. So any individuals can just dump the votes on the server or the CTF in favorable of one candidate participating in the election.

1.2.2 Simplistic voting Protocol # 2

- Each voter signs his vote with his private key.
- Each voter encrypts his signed vote with the public key of a Central Tabulating Facility.
- Each voter sends his vote to the CTF.
- The CTF decrypts the votes, checks the signatures, tabulates them and makes the results public.

The inherent problem associated with this particular voting protocol is that the secrecy is not maintained, that is, the CTF knows who voted for whom which makes this protocol insecure.

1.2.3 Voting with Blind Signatures

- The blind signature protocol disassociates the voter from the vote.
- Each voter generates 10 sets of messages, each containing a valid vote for each possible outcome. Each message also contains a randomly generated identification number, large enough to avoid duplicates with other voters.

- Each voter individually blinds all of the messages, and sends them with their blinding factors, to the CTF.
- The CTF checks the database to make sure the voter has not submitted his blinded votes for signatures previously. It opens nine of 10 sets to check that they are properly formed. Then it individually signs each message in the set. It sends them back to the voter, storing the name of the voter in the database.
- The voter un blinds the messages and is left with a set of votes signed by the CTF.
- The voter chooses one of the votes and encrypts it with the CTF's public key.
- The voter sends his vote in.
- The CTF decrypts the votes, checks the signatures, checks its database for a duplicate identification number, saves the serial number, and tabulates the votes. It publishes the results of the election, along with every serial number and its associate vote.

Pros	Cons
• The votes are unique. • CTF does not know whom any one voted.	• If submitting a vote isn't completely anonymous, the CTF can determine how each voter voted. • Nothing stops the CTF from generating its own signed/valid votes by itself (e.g. the CTF submits false votes). • Voter cannot prove that the serial number in question was actually his.

Table 1.1 Pros and Cons for Voting with Blind Signatures

1.2.4 Voting without a Central Tabulating Facility

- This is a protocol where there is no central facility to take care of the election and the voters them selves form a group and can watch each other.
- Each voter has a public/private key pair, and everyone knows each other's public keys.
- Every voter generates a random string and his vote encrypts and sends to each other.

Pros	Cons
• If any body cheats, it will be known. • It is impossible to figure out who voted for whom. Thus the confidentiality is secure enough.	• Quite Complicated and extremely impractical amount of computation required. Would never work in a real election.

Table 1.2 Pros and Cons for Voting without a Central Tabulating Facilities

1.2.5 Voting with a Single Central Facility

- The CTF publishes a list of all eligible voters.
- Within a specified deadline, each voter tells the CTF whether he intends to vote.
- The CTF publishes a list of voters participating in the election.
- Each voter receives an identification number.
- Each voter generates a public/private key pair and sends the encrypted vote and the random string to the CTF.

Pros	Cons
• Voters can change their votes.	• A corrupt CTF can allocate the votes of those who intended to vote but did not. • Voters cannot verify or prove their vote.

Table 1.3 Pros and Cons for Voting with a Single Central Facility

1.2.6 Voting with Two Central Facilities

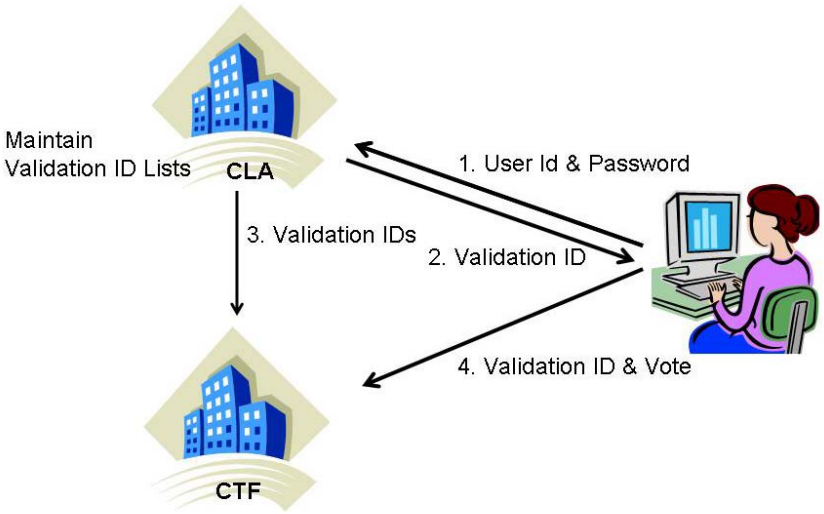

Figure 1.1 voting with Two Central Facilities Communication

- Each voter sends a message to the CLA asking for a validation number.
- The CLA sends the voter back the random validation number. The CLA maintains the list of validation numbers. The CLA also keeps a list of the validation numbers' recipients, incase someone tries to vote twice.
- The CLA sends the list of validation numbers to the CTF.
- Each voter chooses a random identification number. He creates the message with that number, the validation number he received from CLA and his vote. He sends this message to CTF.
- The CTF checks the validation number against the list it received from the CLA. It crosses the validation number if it is there which prevents the voter from voting twice.
- After all the votes are received the CTF publishes the outcome as well as the list of identification numbers and for whom their owners voted.

Pros	Cons

• CTF and CLA watch each other.
• The voter can make sure his vote was counted.
	• The CLA could certify ineligible voters or certify eligible voters multiple times.
	  - **CLA publish a whole list of certified voters.**
	• CLA and CTF work together.
	  - **Make the different database with separate organization control.**
	• Voter cannot verify his vote *(Security Trade off).*
	• Once the user logs in, he has to vote.

Table 1.4 Pros and Cons for voting with Two Central Facilities

2. Design and Implementation

There are several secure voting systems proposed with different protocols; for example, Voting with Blind Signatures, Voting with a Single Central Facility, and Voting with Two Central Facilities as we mentioned above. Each of them has their own advantages and disadvantages; for example, most of them cannot prevent the cheating on the voting server. The server might either ignore to count a vote or count a multiple votes. Also, the voter himself has no mechanism in order to proof if he did vote.

Then, given the benefit of implementing two servers, “The CTF and CLA 'watch' each other -- neither one on their own can cheat without the help of the other.”, although there are few drawbacks for this scheme, there are the obvious solutions to overcome these. As a result, we chose to implement this voting scheme and also modified this in order to overcome all drawbacks and made it possible in practice.

2.1 Protocol constraints

The drawbacks of the Voting with Two Central Facilities protocol [2] are shown below. Given the information from [2] and [5], we modified some of the functionality to solve most of the problems;

1. If a voter determines his vote was tabulated incorrectly, he cannot prove that the ID number in question was actually his.

   Solution: Voter can request the valid validation number from CLA then sends the validation number back to CTF in order to verify if his vote was really his. (Verify Vote function)

2. In case non-eligible voters try to guess the valid validation numbers.

   Solution: we could minimize this threat by making the number of possible validation numbers **much** larger than the actual validation numbers. For example, 100-digit numbers for a million voters (assuming the validation numbers are generated randomly).

3. The CLA could certify ineligible voters or certify eligible voters multiple times.

   Solution: we made the CLA publish a whole list of certified voters (not their validation numbers) For example, it is impossible to have the number of voters on the list less than the number of votes tabulated. However, if some certified voters did not bother voting and in order to prevent CTF cheats, instead of sending the whole validation numbers from CLA, each time the voter requests along with his validation number, the CTF made a request to ask if this is valid validation number from the CLA. Also, CLA will not know who really votes.

4. The CLA and CTF (if working together) could possibly figure out who voted for what by comparing their respective lists.

   Solution: we made the different database with separate organization control.

   Practical Constraint:

5. After the voter logs out or closes the voting session (Voter-CLA), CLA assumes each voter is already voted. So CLA will return the duplicated vote.

Solution: CLA does the authentication process only. CTF does the authorization process; however, each voter request, CTF will forward the message to do the authentication with CLA. Also in order to lure CLA not to record who does vote, every request will forward to CLA not only "vote" message such as requesting candidate and verifying vote.

6. Voter can not come back to verify his selected candidate. (CTF might cheat and vote for somebody else)

   Solution: CTF keeps the record of hash of each validation number who voted and the selected candidate.

2.2 Protocol Design

This section describes our modified protocol from the original Voting with Two Central Facilities protocol [2]. Figure 2.1 shows the protocol communication and Table 2.1 shows an explanation.

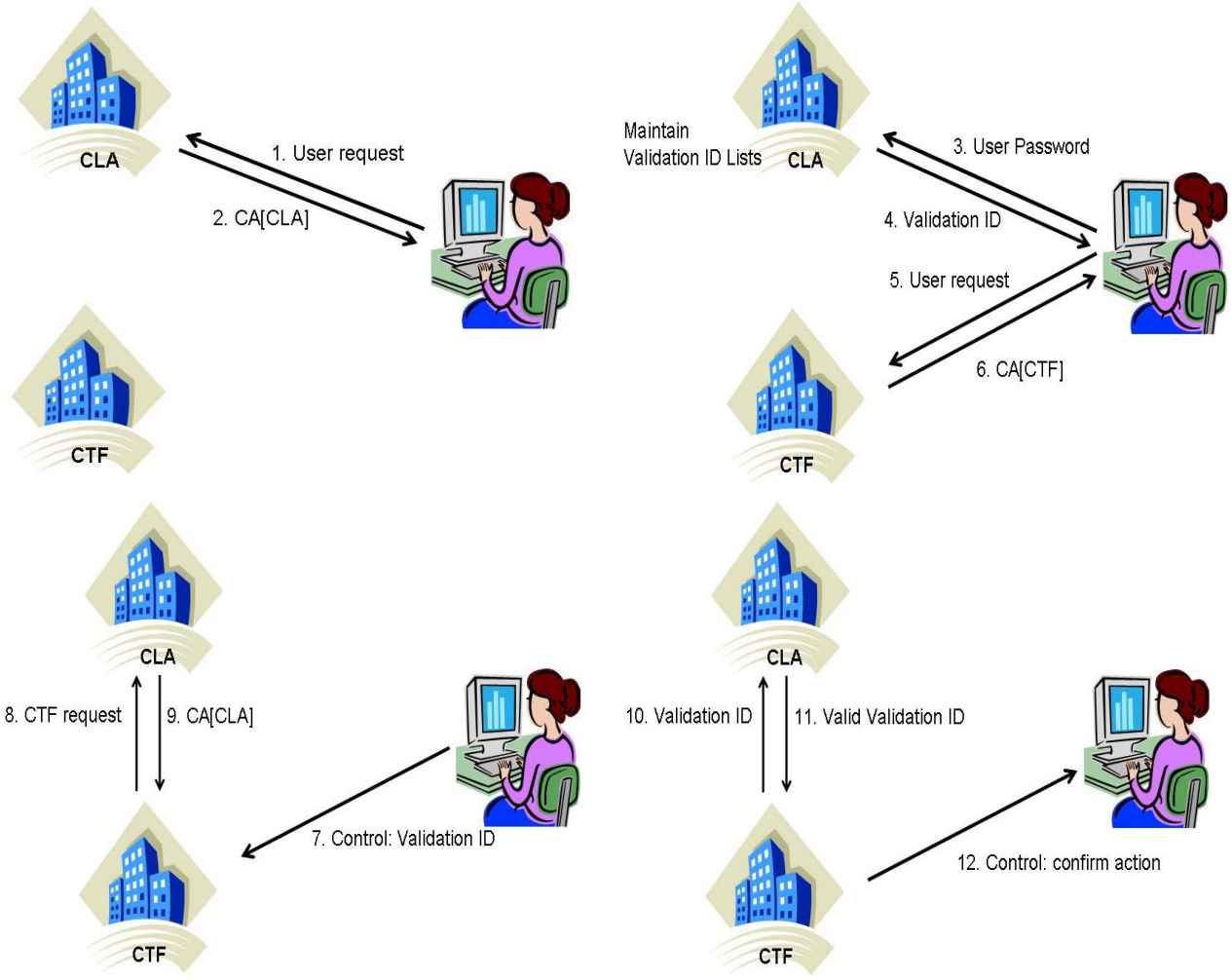

Figure 2.1 Voting with Two Central Facilities scheme

Modified Voting with Two Central Facilities
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 also maintains a list of validation number. The CLA also keeps the list of the validation numbers' recipient in order to be verified by CTF request message. 3. 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 message to the CTF. 4. The CTF checks the validation number by forwarding the message to CLA to verify. If the validation number is valid, the CTF will hash the validation number and keep it in order 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 are received, the CTF publishes the outcome as well as the lists of the identification numbers and for whom their owners voted.

Table 2.1 Modified voting with Two Central Facilities protocol

2.3 Protocol Implementation

Figure 2.1 shows the system overview for the Voting with Two Central Facility scheme. There are mainly three components communicating with each other (Table 2.2). Also, CA authorizer does the extra secure delivery to send user/password, and does CA authorization for CLA and CTF public keys.

Voter	Client/GUI for interaction with voters
CLAServer/ CLAServerCTF	Central Legitimization Agency - Used to authenticate users
CTFServer	Central Tabulating Facility - Used to compile results, process the election, and publish pertinent information

Table 2.2 Voting Components

2.4 System Design

- Platform : Java 1.4 (Platform independent & inbuilt security features)
- GUI : Java Swing
- RSA (1024 bits) : Public Key encryption
- BlowFish (56 bits) : Symmetric Key encryption
- SHA-1 (160 bits): signature/Hashing
- Nonce and Nonce+1 for returning message
- CA : Secure Public Key Transmission

2.4.1 Assumption

- There is another secure connection for delivery "user", "password", and "CA public key" to each user.
- Initially, CTF public and private keys, CLA public and private keys, CA for three agents are predefined.
- This project emulates CA that simply signs a public key with SHA-RSA algorithm. These certificates do not have any expiration dates or the ability to recall.

- Class “GenerateRandomKeys.class” is used to generate (1) and (2).

GenerateRandomKeys.class
Random Key Generator function is used to generate the random keys.

```
byte[] seed = sr.generateSeed(seedByteCount);
            sr = SecureRandom.getInstance("SHA1PRNG");
            sr.setSeed(seed);
```

This class also generates the predefined user, password, and validation ID, “CLA.voters”, the candidate lists, “CTF.candidate”, and also the candidate lists file which used for counting the votes, “CTF.candidate-vote”.

2.5 Agent Communication

2.5.1 Voters<->CLA

- Voters send the request to vote with VoterID to authenticate with CLA. First, CLA send the CA back to Voters. Then Voters decrypt CA with CA public key to get CLA public key. Voters send encrypted message, “CLAID-VoterID”, with CLA public Key encryption back to CLA together with the session key by Blowfish symmetric key.
- CLA decrypts the message and uses Voters’ session keys to communicate and send “CLAID-VoterID” with SHA-1 back to insure the message integrity.
- Voters send user and password to authenticate and then received the validation ID back from CLA

Control	User	Password	ID	Nonce	SHA-1

Control	ValidationID	ID	Nonce+1	SHA-1

Table 2.3 Voters<->CLA authentication protocol (Voters->CLA and CLA->Voters)

Control	RequestValidationID and ReplyValidationID
ValidationID	2132935544077503128/ InvalidID
Nonce/ Nonce+1	4701222 and 4701223
ID	CLAID-VoterID
SHA-1	OAoxfmvT+Bsx+ohnDuFJSjL9TiI=

Table 2.4 Voters<->CLA authentication protocol example

2.5.2 Voters<->CTF

CTF serves three functionalities in order to communicate with Voters; RequestCandidate, RequestMyVote, and VOTE.

2.5.2.1 Users request candidate lists

- After voters authenticate with CLA, they send the request candidate lists with the validation ID to CTF. CTF forwards the message to CLA in order to verify if the validation ID is valid. If it is valid, CTF forms the protocol and sends the lists of candidate to Voters.
- In order to communicate with CTF, Voters do the same process as CLA did. Voters send the request to CTF then CTF sends CA back to Voters. Next, Voters decrypt CA with public key of CA and get CTF public key. Voters do three way handshake with CTF. Then Voters send session key to CTF for later communication.

Control	ValidationID	ID	Nonce	SHA-1

Control	ValidationID	Candidate List	Nonce+1	SHA-1

Table 2.5 Voters<->CTF communication protocol (Voters->CTF and CTF->Voters)

Control	ReqCandidate and ReplyCandidate
ValidationID	21329355440775031 28/ InvalidID
Nonce/ Nonce+1	4830646 and 4830647
ID	VoterID-CTFID
SHA-1	0OdmNEgvZjoPkENeGwarBbFhylA=
Candidate List	Adams,John Carter,Jimmy

Table 2.6 Voters<->CTF communication protocol example

2.5.2.2 Users vote

- After verifying the candidate lists, since Voters can vote once, CTF will forward the message to verify the validation ID. If it is valid, check if it is duplicated vote. If not, cast the vote to the ballot then send the confirm message back to Voters along with the current count and updated count for that candidate.

Control	ValidationID	Candidate	ID	Nonce	SHA-1

Control	ValidationID	Candidate	Current	Update	ID	Nonce+1	SHA-1

Table 2.7 Voters<->CTF communication protocol (Voters->CTF and CTF->Voters)

Control	ReqVote and ReplyVote
ValidationID	21329355440775031 28/ InvalidID/ DupVote
Nonce/ Nonce+1	6065075 and 606507
ID	VoterID-CTFID
SHA-1	ilHy0fHgD3MMfKEcmq9zqwYDugc =
Candidate List	Adams,John
Current/ Update Vote	0/ 1

Table 2.8 Voters<->CTF communication protocol example

2.5.2.3 Users verify their votes

- Voters can verify their votes in order to make sure who they voted for.
- After making verification with CLAServerCTF, CTF hashes the validation ID and compares to the voting record to retrieve the selected candidate and sends it back to Voters.

Control	ValidationID	ID	Nonce	SHA-1

Control	ID	Candidate	Nonce+1	SHA-1

Table 2.9 Voters<->CTF communication protocol (Voters->CTF and CTF->Voters)

Control	ReqmyVote and ReplymyVote
ValidationID	21329355440775031 28/ InvalidID
Nonce/ Nonce+1	5038917 and 5038918
ID	VoterID-CTFID
SHA-1	ZVV62NU8AKyp9qrBmdNyWtA/p8M=
Selected Candidate	Adams,John

Table 2.10 Voters<->CTF communication protocol example

2.5.3 CTF<->CLAServerCTF

- Each request from Voters we mentioned above, CTF will forward the message to CLA in order to verify if the validation ID is valid

Control	ValidationID	ID	Nonce	SHA-1

Control	ID	Candidate	Nonce+1	SHA-1

Table 2.11 CLAServerCTF<-> CTF communication protocol
(CLAServerCTF ->CTF and CTF-> CLAServerCTF)

Control	ReqVerifyValidationID and ReplyVerifyValidationID
ValidationID	21329355440775031 28/ InvalidID
Nonce/ Nonce+1	4434502 and 4434503
ID	CTFID-CLAID
SHA-1	ZVV62NU8AKyp9qrBmdNyWtA/p8M=
Selected Candidate	Adams,John

Table 2.12 CLAServerCTFVoters <-> CTF communication protocol example

3. Experimental results

This section answers the question why this protocol is secure by analyzing the security features of the secure online voting system requirements;

- Only authorized voters can vote.
  - There is a unique random validation ID for each user.
  - It's computationally infeasible for an attacker to guess at a valid pair and also validation ID (the length is long enough).
  - Only CTF can cast a vote with valid validation ID.
- No one can determine for whom anyone else voted.
  - All transaction is secure and singed to prevent someone else to intercept the message.
- No one can vote more than once.
  - Given unique validation ID mapped to pair of user and password, CTF return "Duplicate Vote" if Voter is already voted by hashing the validation before record. So CTF does not know the validation ID but hash of that.
- No one can duplicate anyone else's votes.
  - Given the assumption that there is another secure channel to deliver user and password directly to each voter, nobody would know those from anybody else.
- No one can change anyone else's vote without being discovered.
  - Given random unique validation id and secure and signed transmission, nobody can change anyone else's vote even CTF.
- Every voter can make sure that his vote has been taken into account in the final tabulation.
  - When Voter casts vote, CTF do count for the selected candidate and return the current and updated count back to Voter to insure if his vote has been taken to final round.
- Everyone knows who voted and who didn't (Optional).
  - Given the recorded validation ID from CLA, CTF can provide the list of who voted by hashing validation ID.

4. Build Instruction

All class component is in "OnlineVoting.zip" composed of five java files; CLAServer.java, CLAServerCTF.java, CTFServer.java, UserInterface.java, Voter.java, and GenerateRandomKeys.java. Unzip and compile all those files, there will be six java class plus Voter[i].class (i = 1…n).

Voter	CTF	CLA	CA
PbCA VoterID	PbCA CTFID CA (PrCA[PbCTF])	CA (PrCA[PbCLA]) CLAID	PrCA PbCA

Table 4.1 Initial Values for each agent

Whenever you run GenerateRandomKeys.class, CA Public and Private Keys, CLA Public and Private Keys, CTF Public and Private Keys will be created. Also, initially CLA.voters contains the user/password/validationID lists for testing purpose and CTF.candidate/ CTF.candidate-vote contains the list of the candidate/the list of updated votes. (In order to satisfy the voting requirement: voters can verify vote and everyone knows who vote, "CTF.whovote" also contains the lists of certified voters and the selected candidate (Optional).

User	Password	Validation ID
Jimmy,Mckovit	1511942270351805973	1980617623697238852
Michele,Hooker	1980617623697238852	2128423979834616641
John,Woo	2128423979834616641	-8130546348431878824
Yu,Miler	8130546348431878824	2132935544077503128

Table 4.2 "CLA.voters" example

Candidates	Number of Votes
Adams,John	6
Carter,Jimmy	0
Bush,George	5
Eisenhower,Dwight	10

Table 4.3 "CTF.candidate-vote" example

Hash of Validation ID	The selected candidate
/UFKw9rFZsW5xWlYb4e2IAqtZyE=	Adams,John
LN14apH6HXsYtVnu5w2SKmjx6BA=	Carter,Jimmy
FMgJxtMufcrsiExvBgoIUvUeX1U=	Bush,George

Table 4.4 CTF Record (Hash of Validation ID and the candidate) "CTF.whovote"

Figure 4.1 shows how to run this voting program. First, run CLAServer then run CTFServer and finally run CLAServerCTF. CLA and CTF can run at any server; however, the valid CLA IP address must be the valid CTFServer argument with valid port. Then, after copying all Voter[i].class and CA public key, run Voter with specifying CLA and CTF IP address. After that the Voter screen shot will appear as Figure 4.2. (Each client needs to install Java 1.4)

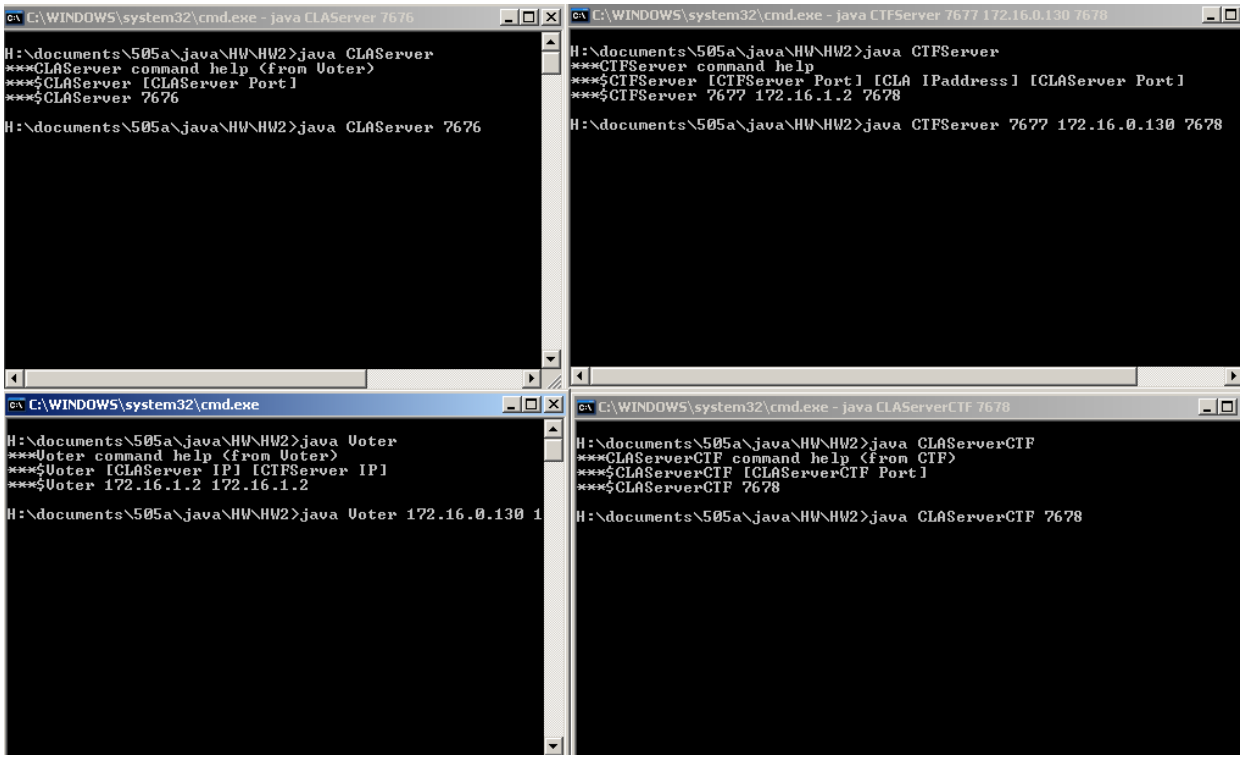

Figure 4.1 How to run voting program example

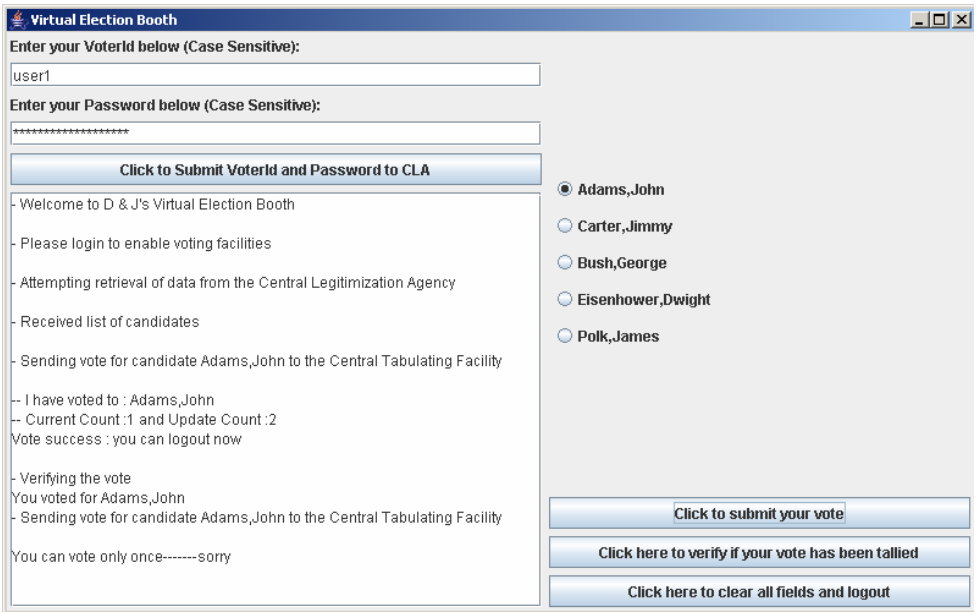

Figure 4.2 voting System Screen Shot
(Authentication, User and Password, Candidate Lists, Vote, Verify Vote, DupVote)

5. Conclusions

Online voting was designed not only to make it convenient for people to vote but also to save the traveling and voting paper based cost. Also due to the today's amazing technology, it can make online voting to be in practice; however, this voting scheme will be never used if we can not maintain the individual privacy and prevent cheating.

In this project, we modified the voting with two central facilities according to [2]. This modified protocol satisfies all secure online voting requirements; however, there are still some security tradeoffs. We did implement the verifying voting functionality but CTF needs to store the validation numbers and the selected candidates.

Although it seems our design protocol is secure enough, this protocol can not trace who buy or sell the votes and it might make easier to do this process by online voting.

6. Recommendations and future work

At first, we tried to implement Java applet but it could not establish the secure connection from the client side (Java Security Limitation) to the server so in this project, the users need to download Voter.class and UserInterface.class to their hosts. However, in order to make the voting system much more convenience, the voter clients may be ported to Java Web Start so each user can access the web based voting system.

In term of protocol design, it would make this protocol much secure if still Voter public and private keys are used to sign each message. Also, to prevent CTF cheating because CTF can see the validation numbers whenever Voters request to communicate. Public key of each Voter should be used to encrypt the message and just make CTF forward the message to CLA then send the verification back to CTF. Due to the time constraint, there are two CLA servers; one is used to communicate with Voter (CLAServer) and the other communicates with CTF (CLAServerCTF). To avoid the system complexity, it is possible to combine those functionalities by threading concept.

Also, since 1024 RSA key was implemented, it can boost the security issue if 2048 bit keys were used instead. Due to the limitation of public key encryption, in 1024 keys scheme only session key was encrypted and sent over the network; however, if 2048 keys was used, we can combine nonce and SHA-1 together with the session keys.

7. References

[1] Stallings William, "*Cryptography and Network Security*," Third Edition, Prentice Hall, 2004.
[2] Schneier Bruce, "*Applied Cryptography*," Second Edition, Jon Wiley & Sons, 1996.
[3] Salomaa, Arto**,** "*Public-key cryptography,"* Second Edition, Berlin New York, Springer, 1996.
[4] Ronald L. Rivest, "*Electronic voting,"* Laboratory for Computer Science , MIT.
[5] Matt Zaske, , "Seth and Matt's Digital voting Presentation" : available online at *http://csci.morris.umn.edu/UMMCSciWiki/bin/view/CSci4554f02/SethMattPresentation*
[6] Dan DuFeu and Jon Harris, *"Online Election System,"* 95.413 Project Report, Carleton University, April 2001.
[7] Java™ Cryptography Extension (JCE) Reference Guide: available online at *http://java.sun.com/j2se/1.4.2/docs/guide/security/jce/JCERefGuide.html*
[8] Java™ Cryptography Architecture API Specification & Reference: available online at *http://java.sun.com/j2se/1.4.2/docs/guide/security/CryptoSpec.html*