

Overview

To design and implement a secure system for electing a student representative for the CS2003 module using the electoral model single non-transferable vote.

Design and Implementation

N.B. - for more information on each method see the Javadocs.

Initially I considered developing the voting system using a web page which the client would access which would, using JavaScript, access a Java web server which would do the majority of the processing and serve interface pages to the user.

However after being told in an email that the entire system had to be implemented in Java, the design was changed to be run through a command line interface on the server side to run the voting system and one or more command line interfaces on the client side to access the server to vote.

An overview of the system can be seen in the provided VotingSystem.png UML diagram.

Class Overview/Summary

- SSLMultiThreadedServer – executed in the terminal on the server side. This class is responsible for setting up an SSLServerSocket and runs the server loop which waits for clients to connect and creates a new thread for each.
- ElectionClient – executed in the terminal on the client side. This class allows the client to connect to the server of the system as a student or as a guest, and displays the results when the election is finished.
- ClientThread – is run by each client connected to the server. Sends the necessary data to the client if they are a guest or student, and sends the final results back to the client.
- ElectoralSystem – once all of the votes have been cast, this class calculates the results by summing the votes of each candidate.
- Voter – represents an individual voter in the system. Allows the client to enter their details to be hashed to validate whether they are an eligible voter, as well checking if the client is a guest and finally allowing them to vote.
- Vote – represents a vote cast by a voter.
- FileIO – handles processing of files including candidates and electoral_register files.
- PieChart_AWT – this class is used to display the final results in the form of a pie chart.
- Ballot – stores the electoral candidates and generates a ballot paper which is sent to the user to allow them to see who they can vote for. Randomizes the order the candidates are displayed in each time a client views the ballot, so no number is associated directly with a single candidate, therefore if a backdoor is somehow discovered it is harder to determine who a voter has voted for, or generate votes for a particular candidate. This video gave me the idea to implement this feature:
https://www.youtube.com/watch?v=izddjAp_N4I
- Crypto – this class is used to hash the client's details as well as the file of voter details loaded in from the electoral_register file. This ensures that no user details are directly stored in the program, providing anonymity when each client votes by comparing their hashed details to the list of hashed details.
- Candidate – Stores the details of each candidate standing for election.

Requirements Satisfied by Solution

a) Only students registered on the module can vote:

When each client runs the program they enter their name, matriculation number, and date of birth. These values are concatenated and hashed. The hash is then sent to the server and compared to a list of hashes produced from the file containing the names, matriculation numbers and dates of birth of registered students (the file is assumed in the specification). If the list contains the hash then the client is allowed to vote as this proves they are registered on the module.

b) Each student has exactly one vote:

As mentioned above, the hash of each client's details are compared to a list of hashes of details of students' registered on the module. When this comparison is carried out, if the client's hash is found in the list, it is removed. This ensures that if they attempt to vote again they cannot.

c) Votes are anonymous - not even the organisers or the software developers can tell which candidate a student has voted for:

The hash method described above protects user anonymity as no user details are stored in the program after the electoral_register file is loaded in. Therefore when a number is received representing a vote for a candidate it cannot be tied to a particular user as hashes cannot be reversed (the electoral_register file would also be deleted after the hash list is produced but for testing purposes it has been kept). The order the candidates are served to the user is also random and so a particular number is harder to use to determine who they have voted for.

d) Votes are secure – this means that users running Wireshark or similar on the LAN cannot understand or replicate message contents or subvert the election in any way:

Secure Sockets Layer (SSL) sockets have been used in the server and client classes in order to provide symmetric encryption when data is transmitted across the network. This ensures that the data will be unintelligible to anyone attempting to observe it on Wireshark or similar. The SSL sockets require certificates to authorise which have been manually signed and placed in a client trust store file and a server key store file. In practice the certificates would be retrieved from an approved certification authority.

e) Voting is secure – fake votes are detected and discarded:

Due to the nature of how the clients are validated anyone attempting to fake a vote would require the name, date of birth and matriculation number of a registered student. The latter of which is difficult to get but not impossible. In reality some people commit election fraud this way by saying they are someone else however the number who do are very few and are often uncovered when the actual person goes to vote. Due to the removal of a hash from the list after a student has voted it could be easily uncovered by the actual student if someone had voted for them.

Also, when the final results are calculated, a UUID is produced using the ID of the current vote being counted. If the new UUID does not equal the current ID of the vote, the vote is automatically discarded as it is presumed to be fraudulent. This check is carried out in the rare possibility that a back door is found to influence how the votes are counted.

Due to the above considerations, the effect of individuals being able to swing the election using fake votes seems negligible.

f) Interaction is as simple as possible using lines of text:

The application was developed to be run in the command line interface of the clients and the server using simple text responses.

g) The results are returned to all voters after the ballot is closed:

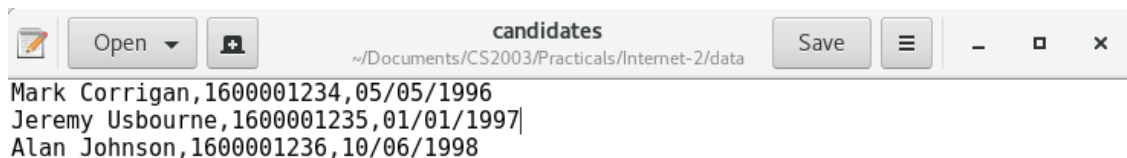
Once the ballot is closed, a pie chart of the results are displayed to any users who connect to the server thereafter. The pie chart contains the names of the candidates, and their number of votes.

h) The ballot will close automatically before the deadline if all possible votes have been received:
In the run method of the ClientThread class there is a condition that checks whether all of the votes have been cast. If this is the case then the results are sent to the user and displayed as the pie chart.

In addition to the above requirements a guest login has been implemented. This was incredibly vague in the specification but has been implemented to return the time left in the election and the number of votes cast so far to the client.

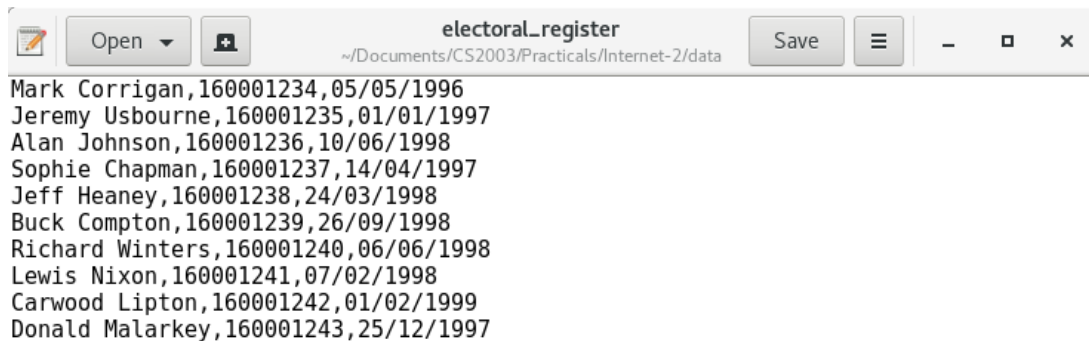
Evidence of Testing

Candidates test file:



```
Mark Corrigan,1600001234,05/05/1996
Jeremy Usbourne,1600001235,01/01/1997
Alan Johnson,1600001236,10/06/1998
```

Electoral register test file:

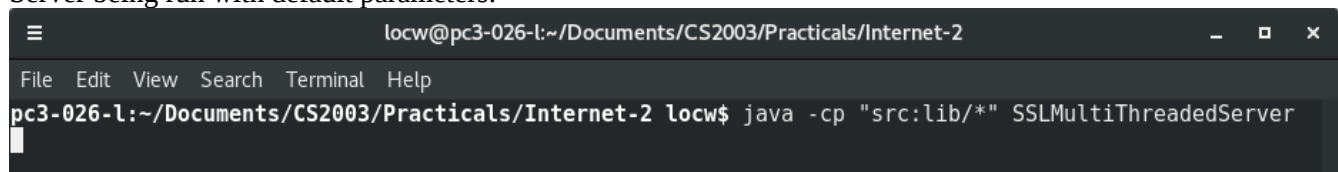


```
Mark Corrigan,1600001234,05/05/1996
Jeremy Usbourne,1600001235,01/01/1997
Alan Johnson,1600001236,10/06/1998
Sophie Chapman,1600001237,14/04/1997
Jeff Heaney,1600001238,24/03/1998
Buck Compton,1600001239,26/09/1998
Richard Winters,1600001240,06/06/1998
Lewis Nixon,1600001241,07/02/1998
Carwood Lipton,1600001242,01/02/1999
Donald Malarkey,1600001243,25/12/1997
```

In the above files it is assumed that candidates can vote. The duration of the election was also set to 10 minutes.

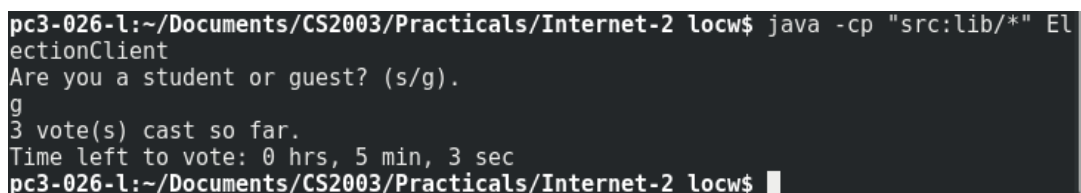
Example output of an election taking place:

Server being run with default parameters:



```
locw@pc3-026-l:~/Documents/CS2003/Practicals/Internet-2
File Edit View Search Terminal Help
pc3-026-l:~/Documents/CS2003/Practicals/Internet-2 locw$ java -cp "src:lib/*" SSLMultiThreadedServer
```

Example of a guest connecting:



```
pc3-026-l:~/Documents/CS2003/Practicals/Internet-2 locw$ java -cp "src:lib/*" ElectionClient
Are you a student or guest? (s/g).
g
3 vote(s) cast so far.
Time left to vote: 0 hrs, 5 min, 3 sec
pc3-026-l:~/Documents/CS2003/Practicals/Internet-2 locw$
```

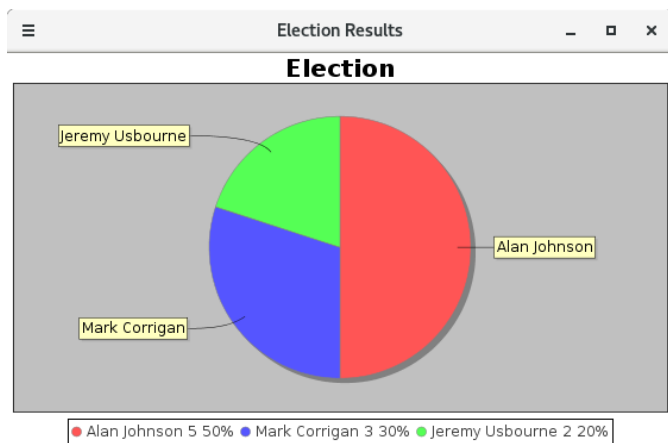
Example of voting:

```
electionClient
Are you a student or guest? (s/g).
s
Please enter your name (as it appears on your matriculation card).
Richard Winters
Please enter your matriculation number.
160001240
Please enter your date of birth (dd/mm/yyyy).
06/06/1998
Server said: You are registered.

Candidate List
Candidate 1: Mark Corrigan
Candidate 2: Alan Johnson
Candidate 3: Jeremy Usbourne

Vote for only one candidate by entering the number of the candidate of your choice:
1
```

Results of election after all ten votes have been cast:



Attempt to monitor vote using Wireshark:

```
0000  00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00  .....E.
0010  00 34 63 e4 40 00 40 06 d8 dd 7f 00 00 01 7f 00  .4c.@. ....
0020  00 01 88 e8 05 52 af ef 52 81 95 31 17 2a 80 10  ....R..R..1.*..
0030  05 80 fe 28 00 00 01 01 08 0a 3a 3f 60 da 3a 3f  ..(....:?'..:?.
0040  60 da
```

As demonstrated by the above screenshot, the data being sent across the server is encrypted and so votes cannot be monitored by malicious individuals on the network.

Example of invalid input:

```
Are you a student or guest? (s/g).
tutor
Entry invalid. Please enter s if you are a registered student or g if you are a guest.
s
Please enter your name (as it appears on your matriculation card).
Bob
Please enter your matriculation number.
1234
Please enter your date of birth (dd/mm/yyyy).
01/01/1940
Server said: Not registered!
Not registered. Closing program.
```

Difficulties Encountered

Initially difficulty were faced when developing SSL sockets as well as multi-threading. These difficulties were overcome by implementing the system using regular sockets and single threading to start off with. I then undertook background reading and research on how to implement both of these features and incorporated them into the developed solution.

Execution of System

To execute the server with/without command line arguments:

Navigate to the Internet-2 folder and execute the command 'java -cp "src:lib/*" SSLMultiThreadedServer'

optionally add parameters 'java -cp "src:lib/*" SSLMultiThreadedServer <port> <keyStore> <key Store password>'

```
pc3-026-l:~/Documents/CS2003/Practicals/Internet-2 locw$ java -cp "src:lib/*" SSLMultiThreadedServer
```

```
pc3-026-l:~/Documents/CS2003/Practicals/Internet-2 locw$ java -cp "src:lib/*" SSLMultiThreadedServer 1362 myKeyStore.jks cs2003student
```

To execute the client side with/without command line arguments:

Navigate to the Internet-2 folder and execute the command 'java -cp "src:lib/*" ElectionClient' optionally add parameters 'java -cp "src:lib/*" ElectionClient <ip address> <port> <trust store> <trust store password>'

```
pc3-026-l:~/Documents/CS2003/Practicals/Internet-2 locw$ java -cp "src:lib/*" ElectionClient
```

```
pc3-026-l:~/Documents/CS2003/Practicals/Internet-2 locw$ java -cp "src:lib/*" ElectionClient 127.0.0.1 1362 myTrustStore.jts cs2003student
```

To run multiple threads simply open additional terminal windows and execute the same command.

Conclusion and Evaluation

To conclude, the implemented solution successfully satisfies the requirements laid out in the practical specification by providing a voting system that is secure, anonymous and able to handle multiple clients.

Given more time I would further ensure tolerance against potentially fraudulent votes as specified in requirement e), however while also attempting to keep the system accessible to use. I also attempted to implement a MapReduce class that would count the votes. The idea being that if the system were used for thousands or millions of votes (such as in a general election) the system would be able to cope. However limitations such as not being able to configure SSL to work with remote servers caused me to abandon the idea. This is something else I would potentially implement given more time.