10-2006

Verifiable Electronic Voting System: An Open Source Solution

Halina Kaminski
University of Western Ontario, hkaminski@csd.uwo.ca

Mark Perry
University of Western Ontario, mperry@uwo.ca

VERIFIABLE ELECTRONIC VOTING SYSTEM: AN OPEN SOURCE SOLUTION

Halina Kaminski* and Mark Perry**
*Dept. Computer Science / **Faculty of Law and Dept. Computer Science
University of Western Ontario, London, Ontario, CANADA
{hkaminski | markp}@csd.uwo.ca

ABSTRACT
Elections, referenda and polls are vital processes for the operation of a modern democracy. They form the mechanism for transferring power from citizens to their representatives. Although some commentators claim that the pencil-and-paper systems used in countries such as Canada and UK are still the best method of avoiding vote-rigging, recent election problems, and the need for faster, better, cheaper vote counting, have stimulated great interest in managing the election process through the use of electronic voting systems. While computer scientists, for the most part, have been warning of the possible perils of such action, vendors have forged ahead with their products, claiming increased security and reliability. Many democracies have adopted electronic systems, and the number of deployed systems is rising. Although the electronic voting process has gained popularity and users, it is a great challenge to provide a reliable system. The existing systems available to perform the election tasks are far from trustworthy. In this paper we describe VEV (Verifiable E-Voting), an electronic voting system which is opne, but also provides for secret and secure voting, and can be used and verified over existing network system.

KEY WORDS
Free Libre Open Source Software; Electronic Voting; Security; Trust

1. Introduction

It is clearly desirable that the operation of our governments be transparent: we need to have trust in the work of our Nation State. In this paper we suggest that the adoption of Free/Libre and Open Source Software (FLOSS) as the primary software resource for key government responsibilities to ensure transparency and trust in such systems, in particular the electoral system,. In this paper we present a Verifiable Electronic Voting system that we have developed and released as FLOSS. [9]

Today, nearly every government in the world wants to know more about free software and how the model works, and the private sector is not far behind. Some governments have already begun the task of migrating to the use of free software in the public sector. The free GNU/Linux operating system now rivals the dominance of Microsoft Windows in controlling how our computers and networks run, at least at an institutional level. For example, the Australian Government Information Management Office's (AGIMO) recognises that the use of open source software is "particularly widespread in areas such as network infrastructure, single-purpose computer servers, security, Internet and intranet applications and network communications" in both the private and public sectors. [10]

In Europe there has been a flurry of projects that are addressing the possibility of widespread adoption of Free Software. The FLOSS project (Free/Libre and Open Source Software) [1] ran from June 2001 for 16 months. It had European Commission funding to gather data on FLOSS use and development. The project was looking to find hard economic data on the effects of FLOSS contributions as a "non-monetary economic network", the distribution patterns of such software, measuring contribution and use, business models, particularly change management.[2] The project was remarkable as the first of its kind to collect such empirical data on a large scale on FLOSS. Following FLOSS's success at making an inroad into providing data on FLOSS, further EU projects have followed. FLOSS-POLS (Free/Libre/Open Source Software: Policy Support) [3] is a current project funded by the European Commission to analyse "government policy towards open source; gender issues in open source; and the efficiency of open source as a system for collaborative problem-solving…. [and] focus on studying the impact of policy and providing policy recommendations."[4] By March 2005, FLOSS-POLS had surveyed 4,138 public authority IT administrators in 13 European member states (excluding Hungary). The key outcomes of this survey are that they found 79% of those surveyed used some FLOSS, and that there is a desire for increased use amongst them. Also notable is the different countries showed different profiles.

An electronic voting system provides the means for the election authorities to carry out the election process using computer-based technology. Although it brings ease to the voters and election candidates, computer scientists argue that voting on line is not safe, because the network, operating system, access, ore even hardware may have security flaws. [5] There has been very little in the way of workable voting system code release as FLOSS.

Many voters already use some sort of computerized voting system. [6] Punch cards, like the ones used in USA presidential elections in 2000, are tallied by a computerized counting machine that detects the punched holes in a ballot. This form of voting has been used since the 1960's [7]. Optical scanners are used for those voting systems that use paper and pen, to detect pen marks made on a ballot. Optical scan vote counters are not as old as punch card technology, but they seem somewhat archaic compared to other technologies that we use everyday. For many people, an electronic voting is the next logical step for elections.

In Brazil and the Netherlands, many voters already use ATM-like machines to cast their vote. Using these machines, voters gather at their traditional voting precinct and cast their ballots in a kiosk, just like the one they have always used. This kiosk retains the privacy that voters want. Voters carry in a cartridge and place it in the e-voting computer, which displays the candidates on a touch-screen, liquid-crystal display. Unlike paper ballots, these machines display information about each candidate aside from their party affiliation, and might even display the candidate's photo so that there is less confusion over identity. A voter makes his choice by touching the screen. Once the voter has completed the ballot, the computer allows the voter to review his or her choice before returning the cartridge to an election official.

In an electronic voting process voters can simply point and click on the candidate they support. This type of voting has the potential to significantly increase voter turnout. In 1998, only 44.9 percent of Americans of voting age took the time to vote. According to [3] many non-voters say that the inconvenience of registering or voting is the main reason they did not cast a ballot.

Dill in [2] points out that unless the voting process is verifiable, it can not be trusted. In most of existing Direct Recording Electronic (DRE) voting machines the internal mechanics of voting are hidden from the voter. It is possible that a computer can easily display one set of votes to the voter while recording entirely different votes in electronic memory. This can be caused by a programming error or by a malicious design of the system. Almost all of DRE machines currently used in USA require official certification, but the election officers are powerless to prevent programming errors in recording of the votes. DRE code is usually protected by code secrecy agreements, so no one but the manufacturer has access to it. Dill et al. says that the only way to have a trusted electronic voting system is to include a voter-verifiable paper audit trail in all DRE voting machines. The voting system that is described in this report provides a secure way to count and to verify the votes.

While computer scientists and critics in North America are concerned with the insecurity of electronic voting machines, Australians designed a voting system five years ago, and they made most of the source code of the underlying software available to the public [8].The system had to be implemented over a secure network (using independent connections). This requirement alone provides enormous difficulty to the election authorities in Australia. The state was able to test only 80 machines distributed among 8 polling stations in their 2002 election.

2. The Task

The design of a "good" voting system, whether electronic or using traditional paper ballots or mechanical devices must be robust against a wide variety of potentially fraudulent behavior. The following voting system requirements were born out of the desire to create a product that would allow modern computer-based technology to truly emulate the secure properties as valued in the public voting. The purpose of this project is to make it impossible for voting authorities to engage in a fraudulent behaviour, and at the same time the system will provide the secrecy for the voters. VEV has been an attempt to provide a voting system that would be:

1. *anonymous* - no one should be able to determine how any individual voted
2. *secret* - all cast votes should be unknown until the election ends
3. *correct* - It should not be possible for a vote to be altered, or for a valid vote to be eliminated from the final tally, or for an invalid vote to be counted in the final tally
4. *honest* - no one should be able to vote twice or change the vote of another voter
5. *public* - *after an election all results should publicly known, but the connection between votes and the voters should be both unprovable and unknown*

Critics of the electronic voting systems say that the voters who use them have no way to verify that their votes are being recorded and counted accurately. In case of an electronic system the only known solution to this problem is to introduce a "voter-verifiable audit trail". Most commonly, this is achieved by adding a printer to the voting terminal. When the voter finishes selecting candidates, a ballot is printed on paper and presented to the voter. If the printed ballot reflects the voter's intent, the ballot is saved for future reference [2].

The design of a VEV helps to overcome these difficulties. In part, the system uses the idea introduced in [9]. It provides a significant improvement to the process of electronic voting by publishing the voting results to the screen. The system does not involve printers or paper receipts. Every user will be able to see the number of votes that were cast and the final results for each candidate, but only the particular user will know if his/her vote was counted and if it was counted correctly.

3. The protocol

There have been a number of conditions that have to be met in order to provide voters with a secure electronic voting system. This paper includes the description of the general steps that needed to be taken in the design of the system to provide the user with voting security. The requirements for the system are the following:

1. Voting takes place over a computer network

The electronic voting system is designed to be implemented and used over an existing computer network. The system includes three major parts: server's side program, client's (voter) side application and administrator's (administrative user) side software. The server-application should be stored and executed on the main network computer. The client-application could be located either on the main computer or on every network's terminal. It is recommended that, for the security reasons, the administrator's application should be stored on the removable storage device (such as floppy, CD), kept secure, and run only when changes are being made to the voting procedure. The administrator's software should be used with extra caution.

2. Only authorized voters can vote

Every voter will be assigned a user's name and a unique password. The administrator will be responsible for choosing the appropriate values for the name and password, since it depends on the election importance as well as the election settings, (e.g. students at the Computer Science Department might be assigned departmental user's names and their students numbers as passwords; the secret service workers can use randomly generated numbers as their identification). The administrator has to deliver the user's names and passwords to each eligible voter. It is up to the election administrators to decide what means of delivery will be chosen (e-mail, regular mail, in person). It is assumed that this is done in a very secure way.

3. The voter can cast only one vote

It is important for the system that it allows each voter to cast one and only one vote. It is required by the democratic election process that there can not be more votes cast than there are voters. This system will provide the option of a re-vote to each user. Therefore it becomes of great importance that the previous vote cast by the particular voter will be erased when that user votes again.

4. Only the voter can know his/her vote

In democratic elections only the voter can know his voting strategy: This is the secrecy requirement. There can not be a trace left between the voter and the vote and all the links should disappear. No one should be able to recognize the voter by looking at the ballot.

5. Each voter can check if his/her vote was counted

There is a great improvement to the electronic voting process in VEV. Every user can check if his/her vote is in the ballot (which means it has been counted). The system will provide the option to check the votes (check the ballot), and the voting strategy identification (discussed below) will be displayed. Each user can count the votes that were cast. He can recognize his vote among the displayed votes.

6. Each voter can change his mind

When the election process progresses, the voter can become aware that he did not vote for the candidate he wanted. VEV provides the option to re-vote. It is assumed the re-vote is available before the final voting date. The user can change his mind multiple times. The system supports a multiple re-vote function. Every time the new vote is cast the existing vote from that user is erased.

The voting system that is described in this paper uses the public-private key paradigm to encrypt information. In VEV, the user's identification number *(id)* and the voting strategy number (v) (which is a numeral representation of the candidate's name) are the two prime numbers that are being used. There are three different algorithms designed to do calculations with these two prime numbers and returning one large number as a result. It is randomly chosen in the program which of the three algorithms is used when the voting is performed.

4. The algorithms

***Function 1*:**
First function uses multiplication function as the underlying calculation. As a result, the product of two prime numbers is returned.

***Function 2*:**
This function calculates the product of two prime numbers. It swaps the values of the individual bytes within the binary representation of the product (namely copies the value of last byte into the byte before the last, and the value of the second last byte into the last byte. The same swapping operation is done to the third and fourth last byte of the product).

***Function 3*:**
This function calculates the product of two prime numbers. It flips (replaces with the complementary value) the values of the individual bits within the binary representation of the product. The algorithm changes the values of bit positions: 3,6,7,12,15.

The fact that both of the prime numbers are randomly generated for each user and for each voting strategy provides enormous security for the system. The standard RSA cryptosystem uses the same p and q throughout its lifetime where in the voting system presented here the probability that the same two numbers will be used twice is very close to zero. The major part of the private key

constitutes the fact that there exists a system defined index that uniquely identifies each candidate. Even if the intruder is able to factor the voting strategy function result, having two prime numbers would not give him any reasonable answer. The secret lies in the knowledge of indexing the candidates and having the function inverses. For this particular reason the usage of 25-bit long prime numbers provides sufficient security to the voting system. The prime numbers are being generated using the constructor for *BigInteger* class from the Java programming language library. The method returns a randomly chosen, 25-bit long positive integer which is a prime number. The probability that the newly generated number represents a prime number will exceed $(1 - 1/2^{100})$. The execution time of this constructor is proportional to the value of the probability parameter (which in this case is 100). In addition, each newly created number is checked once again by isPrime() function from the Java class library.

In the remainder of this paper, the word user will be used interchangeably with the word voter to describe the person who is casting the vote. The word server will be used to describe the software implemented and executed on the network's main computer and the word client constitutes to the computer program that provides the graphical interface to the user, and allows for communication between the server and the voter.

Phase 1: Preparation

VEV (hereafter called system) publishes the number of eligible voters and the deadline for the response. In order to be able to vote, each voter has to confirm his intention to vote and only those who respond will be allowed to cast the vote later. There will be a specified period of time when the voters can respond.

Phase 2: Voting Scenario

When the date for the user's response passes, the system enters the phase of the main voting process. The voting system running on the server is constantly waiting for the user to connect. The voter starts using the system by entering his user name and the password which he previously obtained from the system's administrator. Then the system authenticates the user. If the system recognizes the user it makes all functionality available to this person (such as vote, re-vote or view the existing votes). If the voter is not a recognized person (either the user's name or the password does not match the records) he is treated as a guest to the system and the only thing that he is able to view are the existing votes. If the recognized user chooses to cast the vote for the first time the system creates the identification number for that user. When the eligible user wants to cast a vote for the first time the client software will randomly generate a 25-bit long prime number (*id*) which will be used to uniquely identify that particular user (that number has to be checked against existing identification tags that have been stored already in the server's database; if such a number exists already, a new identification tag is generated).

In the next step of casting a vote the user chooses the candidate that he wants to vote for. The system displays the names of the election candidates and the user chooses one of them. The numerical encoding for every voting strategy (e.g. name of candidate) is a large prime number. VEV is able to handle as many as 24 candidates to be voted for. The number 24 provides the opportunity for the unique encryption of each voting strategy. First, all numbers that end with 1,3,7,9 between 10 and 100 are selected (the underlying reason for that is the fact that the prime numbers end with 1, 3, 7, 9). This way a set of two digit numbers has been created (hereafter called indexes). For every index from the set, an election candidate is assigned. When the user chooses to cast a vote for a particular candidate, a random 25-bit prime number (v) is generated such that the first digit is equal to the first digit of the index and the last digit of v is the same as the last digit of an index. E.g.: Say we have an election candidate Anna S. Initially the system had assigned an index identification number to her that is 51. If the voter decides to cast the vote for Anna S. the client's program will randomly generate the prime number 5…..1 (first and last digit match the index).

Next, the user sends the pair of integers (id, f (id, v)) to the system where f is a randomly chosen encryption function (one of three algorithms that are explained above); id is the identification tag generated for the user, and v is the candidate's name represented in the number; f (id, v) is the result of the encrypting method that takes id and v as its parameters. The system does not know the connection between the user's name and the id tag (or the voting strategy). The only association that is known to the system is the connection between the id tag and the vote function f (id, v). The user is asked to write down his identification number (id) and the result of the voting strategy function. He is also informed by the system to keep these numbers secret. When the server's side receives the numbers it publishes the voting function result to the screen. The user can easily check by choosing the *Check Votes* option if his vote was counted.

For each election candidate the system displays f(id,v) to the screen. This way the user can check the correctness of his vote and the distribution of all votes. Publishing the voting strategy will serve an additional function. Every election candidate will be able to check if the votes were counted correctly. It might be of a great importance for the candidates, because it is known that the elections have been won by a difference of several votes.

5. Implementation

The primary advantage of public-key cryptography is increased security and convenience. The private key never

needs to be transmitted or revealed to anyone. This section explains the major steps that have to be taken in order to implement this voting system whose security is based on the usage of public-private key paradigm. It has been assumed that the server is running on the main computer and is constantly waiting for a client to connect. It is also anticipated that every user possesses the knowledge of his user's name and password. The italic type characters will be used to indicate the processes occurring on the server's side of the voting system.

6. Main server's functionality

Pseudo code:

```
//second phase: user can vote and re-vote
```

The system recognizes the user and chooses the correct response depending on user's input:

```
            …
            switch (state) {
case VOTE:// user wants to vote
       create new Vote object
       outLine = "User wants to vote";
       os.println(outLine);
       //check once more if the user can vote

         while(true){
         read the input from the client
         get the id (prime) from the client's system;
         }
         inString=is.readLine();
         BigInteger t = new BigInteger(inString);
         hash_function = (HashFunc)Oin.readObject();
    //set Vote object variables:
         current.setId(prime);
         current.setVoteFunction(t1);
         current.setHash(hash_function);
         voteStructures.addElement(current);
                    //update the users parameters
         writeVoteFile();
         found.setVotedOnce(true);
         writeUsersFile();
         outLine ="Voting done successfully.";
       }
       case PRINTRESULTS:
       // user wants to print results
       outLine = "The following are the voting results:";
       os.println(outLine);
       readVoteFile();
…
       case VOTEAGAIN:
       // User wants to re-vote
       Create new Vote object
       readVoteFile();
       toChange = this.findPrimeVote(id);//find an existing vote
//change the voting strategy
                    …
      toChange.setVoteFunction(hash_vote);
      toChange.setHash(hash_function);
```

Step 1: Authentication

1. Voter starts the execution of the client's side program.
2. Client asks the user to enter his name.
3. *Server's side application checks if the name exists on the list of users that are eligible to vote.*
4. If the name exists, the user is asked to enter his password; otherwise the user is considered to be system's guest.
5. In case that the user's name exists, the server checks if the password matches the user's name (if the password does not match, the user is considered to be the guest).

Step 2: User's operations

Phase 1 (the time allocated to acknowledge users' responses with the willingness to vote)

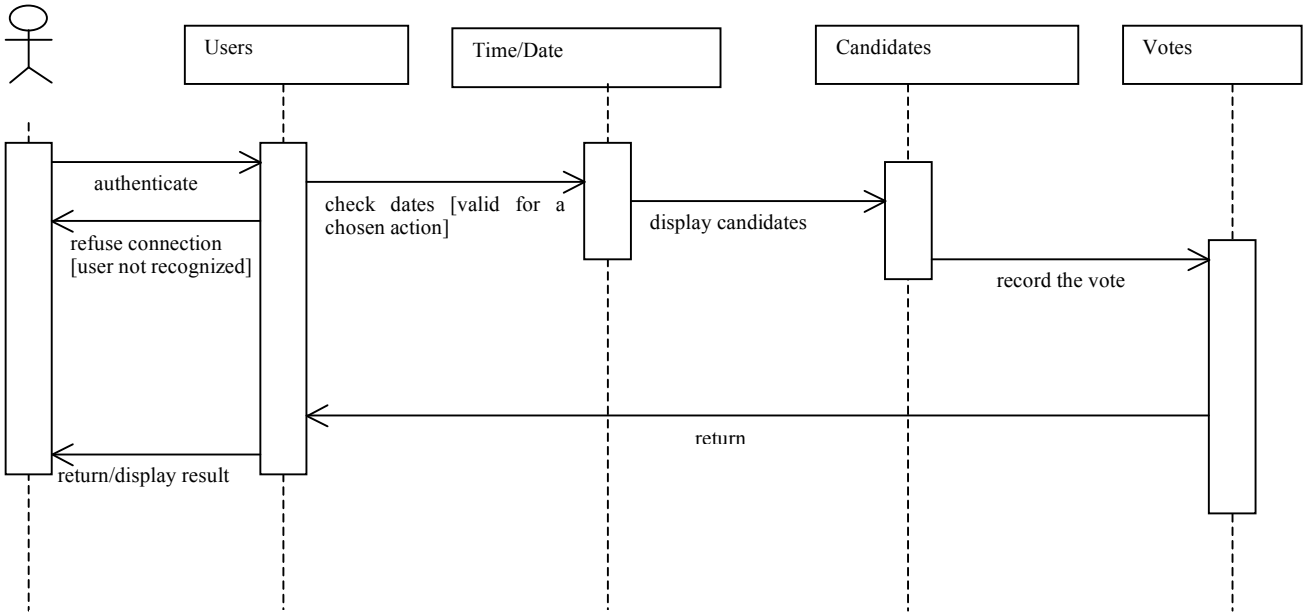
1. Client displays the names of the users that are eligible to cast a vote.
2. User chooses the option to confirm voting or the option to exit.
3. *If user chooses to confirm voting the server records user's willingness to vote.*
4. Client displays the "Thank you" message and informs the user about voting dates.

Phase 2 (the time allocated for the actual voting)

User chooses to vote

1. Server checks (using user's name and password) if the user has voted already.

- If the user did not cast his vote yet, the client randomly generates 25-bit long prime number and assigns it as an identification number to that particular voter.
- The message is displayed on the screen asking the user to take a note of this number and not to reveal it to anyone.
- Client displays the names of the election candidates, and asks the user to choose one of them.
- User types in the number of the candidate for whom he wants to cast the vote.
- Client randomly generates 25-bit long prime number called voting strategy, such that it meets the specification to match the first and the last digit with candidate's index number (private key requirement).
- Client performs one of the encrypting functions (called also a voting function; there is a random choice made to use one of the three available encrypting methods) on the user's identification ta g and the voting strategy number.
- Client displays the voting function to the user. The user is asked to write the number down and to keep it confidential.
- Client sends the pair (identification tag, voting function result) to the server.
- Server stores the vote information in its database



Sequence Diagram for Voting Scenario

- Server records that the user voted already. It is done to prevent the user from casting multiple votes.
- When the user chooses to exit, client disconnects and the link between user's name and his vote disappears.

2. If the user previously cast the vote, he is asked to choose the re-vote option

User chooses to re-vote

1. Client asks the user for his identification tag number.
2. Client asks the user for his voting function.
3. *Server checks if the vote exists.*
- Client displays the names of the election candidates, and asks the user to choose one of them.
- User types in the number of the candidate for whom he wants to cast the new vote.

Client randomly generates 25-bit long voting strategy number, such that it meets the specification to match the first and the last digit with the candidate's index number (private key requirement).
- Client performs one of the encrypting functions (there is a random choice made to use one of the three available encrypting methods) on the user's identification tag and the voting strategy number.
- Client displays the result of voting function to the user. The user is asked to write the number down and to keep it confidential.
- Client sends the pair (identification tag, voting function result) to the server.
- *Server stores new vote in its database and erases the old vote.*

User chooses to check the votes

1. Client displays the voting functions for all votes that were cast. To make it easier for the user to find his vote, the list of votes is displayed in an ascending order.
2. User can check if his vote is in the ballot, which means it was counted.

User chooses to exit

1. Client displays the "Goodbye" message
2. Client disconnects from the server

Phase 3 (after voting deadline)

1. Client displays all the voting functions to the screen. The votes are displayed in such a way (see below), that for every candidate the voting function numbers are displayed in an ascending order. The user can check if his vote was counted correctly, and the election candidates can verify the voting results.

7. Conclusions

Voting software cannot be treated in the same way as a word processor or other applications, as we have even less reason to blindly trust the vendor – especially when the whole country's future is at stake. Most of the recent news about harnessing electronics for the election process has been bad.[8] While much work in the USA is aimed at strengthening the ever-tight security around the software source code (it has been suggested that the voting application source code could not be reviewed even if challenged in court), there is a contrary approach whereby the voting code is made public, ie released as FLOSS. It is often argued [e.g. 10], that the only way to have a trustworthy system is to open the source code of cryptographic functions to the public. The algorithm can really be considered secure when is examined by many experts. "… [t]he only way to have any confidence in an algorithm's security is to have experts examine it."[11]

Australian officials believe that elections can benefit from involving the voters in the software development process. Perhaps a truly open system can alleviate some of these issues.[9]The voters can dictate the requirements including security and functionality of the voting system. No matter how many election flaws are found, and despite their severity, electronic voting systems are here to stay and serve us all. The only question remains: "How much, or little, trust can we afford?"

8. Acknowledgements

This research was funded by Natural Science and Engineering Research Council. Thanks to Shiva Mohan for coding support, and Professor Lila Kari for algorithm development.

1. References

1 < http:// http://flossproject.org/ >. The project refers throughout to OS/F software as in "open source/free'.

2 ibid.

3 General description of project at < http://flosspols.org/ >

4 The survey report is at < http://flosspols.org/deliverables/FLOSSPOLS-D03%20local%20governments%20survey%20reportFINAL.pdf > and was completed on 14 July 2005.

[5] Dill, David, Schneider Bruce, Simons Barbara *Voting and Technology: Who Gets to Count Your Vote?* Communications of the ACM, August 2003.

[6] Kohno, Tadayoshi; Stubblefield, Adam; Rubin, Aviel D.; Dan S. Wallach. *Analysis of an electronic Voting System*, IEEE Computer Society Press, Johns Hopkins University Information Security Institute Technical Report TR-2003-19, 2003.

[7] Neumann Peter, Mercuri Rebecca, Weinstein Lauren. *Internet and Electronic Voting, The Risks Digest*, ACM Committee on Computers and Public Policy, Volume 21, Issue 14, December 12, 2000.

[8] Zetter, Kim. *Aussies Do It Right: E-Voting*, Wired News Magazine (online), November 03, 2003.

[9] H.Nurmi, A.Salomaa, L.Santean. *Secret ballot elections in computer networks*. Computers and Security, nr.10, 1991, pp.553-560.

[10] Bruce Schneier is an internationally renowned security technologist and author. Schneier is best known as a security critic and commentator. His books include: *Applied Cryptography*, *Secrets and Lies*, and *Beyond Fair*.

[11] Schneier, B. Crypto-Gram Newsletter, September 15, 1999, Retrieved on August 26, 2004 from: < http://www.schneier.com/crypto-gram-9909.html >.

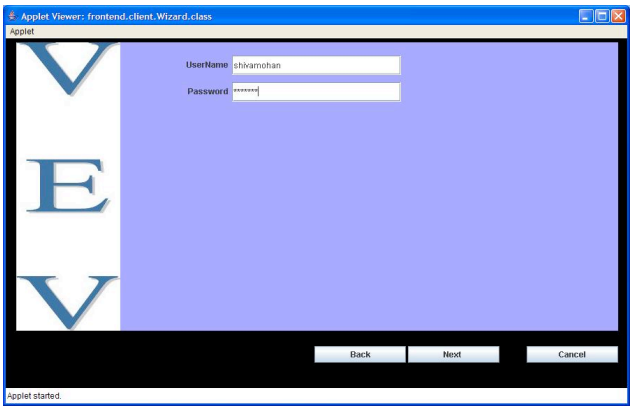
[12] See Brennan Center Task Force on Voting System Report *The Machinery of Democracy: Protecting Elections in an Electronic World* 27 June 2006

[13]The VEV software is open: < http://sourceforge.net/projects/vev >.

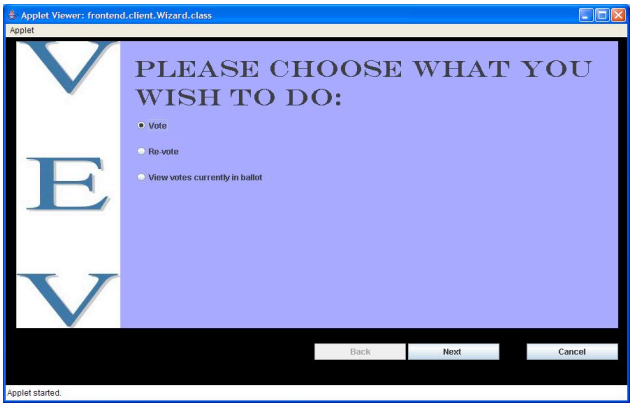
[14] AGIMO, A Guide to Open Source Software (2005), p.10.

User Screens

Authentication



User Choice



Results display