Practical Aspects of DynaVote e-Voting Protocol

Orhan Cetinkaya[1] and Mehmet Levent Koc[2]
[1]METU, Ankara, Turkey
[2]Bilkent University, University of Winsconsin-Madison, USA
corhan@ceng.metu.edu.tr
koc@cs.wisc.edu

Abstract: Voting is regarded as one of the most effective methods for individuals to express their opinions on a given topic. Electronic voting (e-Voting) refers to the use of computers or computerised voting equipments to cast ballots in an election. e-Voting performed over Internet can be universally accepted in the upcoming years due to the fact that Internet plays a key role in people's lives. The DynaVote e-Voting protocol claims that it is practical over a network since it does not use complex algorithms and has no physical assumptions such as untappable channels, whereas fulfilling core voting requirements such as privacy, accuracy, uncoercibility and individual verifiability. Software development requires a considerable amount of time and money. Therefore, in order to utilise all resources, the prototype implementation gains more importance as it gives quick feedback about the practicality of the system. This paper presents a prototype implementation of DynaVote e-Voting protocol over the Internet. Since DynaVote relies on Pseudo-Voter Identity (PVID) scheme, which is an unlinkable pseudo identity mechanism, the prototype includes implementation of a PVID scheme component as well. The main outcome of this study is to prove that DynaVote protocol over Internet is practical and applicable in real life and to illustrate that PVID scheme provides unlinkability. This study also contributes some improvements in DynaVote e-voting protocol. Furthermore, this paper analyses how the prototype fulfils some electronic voting system requirements such as efficiency, transparency and mobility.

Keywords: DynaVote, e-Voting, electronic voting, implementation, practically, e-Voting requirements

1. Introduction

Electronic democracy is a necessity in this era of computers and information technology. Electronic voting (e-Voting) is one of the pillars of the e-democracy, which refers to the use of computers or computerised voting equipment to cast and to tabulate ballots in an election in a trustable manner. Due to the nature of electronic systems the security and reliability of the system should be handled properly in order to make the e-Voting system an applicable alternative to the paper based voting system for the governmental elections.

The DynaVote voting protocol claims that it is secure and practical over a network whereas fulfilling core voting requirements (privacy, eligibility, accuracy etc.) (Cetinkaya 2007-2). DynaVote does not use complex algorithms such as homomorphic encryption and does not require anonymous communication channels such as mix-nets. Besides it has no physical assumption such as untappable channels. It only needs an unlinkable pseudo identity mechanism, so it employs Pseudo-Voter Identity (PVID) scheme (Cetinkaya 2007-1) which relies on blind signature. PVID scheme provides PVIDs that are unlinkable to the voter's real identity.

It is widely known that the development of whole protocol as a software implementation is not an easy task. Developing software requires considerable time and cost. Therefore, in order to utilize all resources, implementation of prototypes gains more importance. This paper presents a prototype implementation of DynaVote voting protocol over Internet. The prototype includes implementation of a PVID scheme component as well. In its current state, the prototype mainly serves experimental purposes to test the DynaVote protocol and PVID scheme. During the development some implementation issues have arisen. Therefore, we have made three improvements in the DynaVote protocol; these are described in Section 3. This study shows that the implementation issues are resolvable and DynaVote protocol over Internet is practical and applicable for large scale elections, as well as illustrating that PVID scheme provides unlinkability.

The remainder of the paper is organized as follows. In the next section, related work is stated. Section 3 illustrates the improved DynaVote protocol. Section 4 explains the details of the DynaVote prototype implementation, analyses how the prototype fulfils some electronic voting system requirements such as efficiency, transparency and mobility and contains the experimental results of the prototype as well. Finally, conclusions are drawn and future work is suggested.

Reference this paper as:
Cetinkaya, O, and Koc, M, L. "Practical Aspects of DynaVote e-Voting Protocol." *Electronic Journal of e-Government* Volume 7 Issue 4 2009, (pp327 - 338), available online at www.ejeg.com

2. Related work

There are numerous cryptographic voting protocols in the literature. The protocols proposed so far could be classified into three categories by their privacy preserving approaches as: protocols using mix-nets, protocols using homomorphic encryption and protocols using blind signature (Forsythe 2005). However, most of them are not practical and not applicable over the Internet (Sampigethaya 2006).

In homomorphic encryption based voting protocols, the voting result is obtained from the accumulation of encrypted votes whereas no individual ballot is opened and the corresponding individual vote remains secret. Correctness of the tallying cannot be guaranteed without validation, so each vote must be verified to be valid in homomorphic voting. One of the implementation studies has been done by Forsythe (Forsythe 2005) and Weber (Weber 2006), where they implemented prototypes for their academic studies.

Ballot tabulations are efficient when the number of candidates is small in homomorphic encryption based voting protocols, however, communication complexity of these protocols is quite high if the number of candidates is large. Computational and communicational cost for the proof and verification of vote validity is relatively high so that homomorphic voting actually becomes inefficient for large scale elections. Thus, voting protocols based on homomorphic encryption are far from being both secure and practical in real-life.

In general, the voting protocols, stating that they satisfy practicality and privacy, have a strong assumption of anonymous communication channels which are provided by verifiable mix-nets. There are several approaches to achieve mix-nets; the main idea is to permute and shuffle the messages in order to hide the relation between the message and its sender. In verifiable mix-nets, a mix server additionally has to prove in zero knowledge that it decrypts/re-encrypts and shuffles the inputs correctly. However, mix servers suffer from computational cost for proving that their mixing is correct, so the main difficulty is efficiency of proof techniques.

Blind signature based voting protocols employ blind signatures in different stages of the voting process in order to assure voter privacy. The idea behind these protocols is that the voter prepares a ballot stating for whom he wishes to vote. He either obtains an authorized ballot or interacts with an authentication authority to make his vote authorized. Finally, he sends his cast ballot to another authority that is responsible for counting votes. Blind signature based voting protocols generally assume the existence of an anonymous channel.

There are several implementations that have been piloted in small scale elections. The SENSUS system (Cranor 1997) was the first to be implemented. The EVOX system (Herschberg 1997) was used at MIT for undergraduate association elections. DuRette (DuRette 1999) improved the EVOX system in order to eliminate single entities capable of corrupting the election. Both DuRette's system and EVOX are very sensitive to failures in communication or servers, these problems were solved by REVS which is proposed by Joaquim et al. (Joaquim 2003) as another implementation based on DuRette's work. Later, some improvements were done on REVS to make it more robust (Lebre 2004). Votopia project (Kim 2002) which is created jointly by Korean and Japanese was tested in the election of the MVP (Most Valuable Player) in the Soccer World Cup of 2002. Votopia is not publicly available and does not provide anonymity. All aforementioned protocols use mix-nets or assume the existence of anonymous channels.

3. DynaVote implementation issues and improvements

During the implementation studies, a detailed analysis of DynaVote protocol revealed a privacy issue. In the original protocol, there is a possibility that corrupted Counter authority may trace the voter's IP over Internet. Thus an improvement is made in the protocol by introducing the Collector authority to distribute the power of Counter. In the improved version of DynaVote, Counter authority has no direct relation with other authorities and the voter.

Another improvement is introducing a *sequence number* which shows the number of dynamic ballot requests for a particular voter. Each dynamic ballot is associated with a sequence number. This improvement is done to handle the recasts in the counting stage more easily and efficiently whereas the original protocol uses *DateTime* information to order the recast votes.

Furthermore, a customization is made in order to increase client-server communication speed. While communicating the servers, DES symmetric key algorithm is used to encrypt the network messages instead of RSA without sacrificing the security of the original protocol.

3.1 Improved DynaVote overview

In this section we will give a brief overview of improved DynaVote voting protocol. A detailed explanation can be found in the original work (Cetinkaya 2007-2). DynaVote protocol consists of three distinct stages: Authentication & Authorisation (performed at the beginning of the election), Voting (carried out during the election period) and Counting (performed at the end of the election). There are five authorities in the protocol:

- *PVID Authority* provides PVID-list which consists of pseudo voter identities which are unlinkable to voter's real identity.
- *Ballot* Generator is an authority which provides dynamic ballot to the voter for each voting operation.
- *Key Generator* provides vote encryption key to the voter via Ballot Generator.
- *Collector* is an authority which collects cast dynamic votes.
- *Counter* reveals and counts the actual votes at the end of the election and then announces the election results.

Interaction between authorities is shown in Figure 1. Counter authority has no direct relation with other authorities and the voter; therefore it is not shown in the figure.

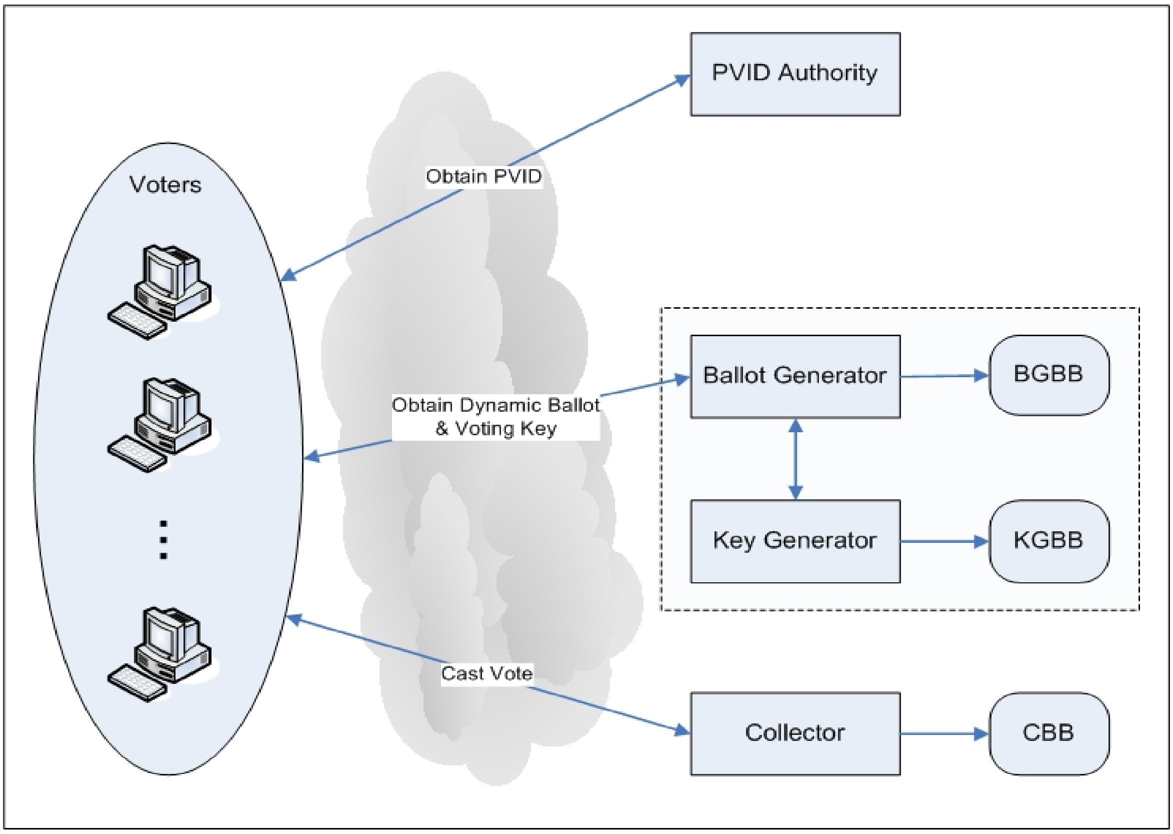


Figure 1: DynaVote overview

3.1.1 Authentication and authorisation stage

In this stage, PVID scheme (Cetinkaya 2007-1) is employed. The voter applies PVID authority to obtain a PVID-list by using his real registration identity (RegID). PVID Authority checks voter eligibility and issues the voter's PVID-list. A PVID-list is nothing but a list of approved anonymous pseudo identities which are unlinkable to the voter's registration identity. In PVID scheme, the voter performs RSA blind signature with PVID Authority in order to obtain PVID-list.

A PVID-list can be used at any time and place during the election period. DynaVote employs PVID scheme for two identities. The voter's real registration identity is hidden to the voting authorities. Authorities can easily check the validity of any PVID by employing PVID Authority's public key on it.

3.1.2 Voting stage

In voting stage the voter obtains a dynamic ballot and casts his dynamic vote. In dynamic ballots, the ordering of candidates in the ballot is dynamically created and changes randomly for each voter. A dynamic vote is defined as the voter's candidate selection in the dynamic ballot. So, dynamic vote has a contextual meaning depending on the ordering of candidates in the dynamic ballot.

The voting stage consists of two phases. In the ballot obtaining phase Ballot Generator provides dynamic ballot to the voter. Key Generator provides vote encryption key to the voter over Ballot Generator. In the vote casting phase, the voter selects his vote from the dynamic ballot and then encrypts his dynamic vote by using vote encryption key. Lastly, the voter casts his encrypted dynamic vote by using his PVID. During the voting stage, Ballot Generator, Key Generator and Collector publish hash of subsets of relevant information on the bulletin boards. Figure 2 represents the overview of the voting stage.

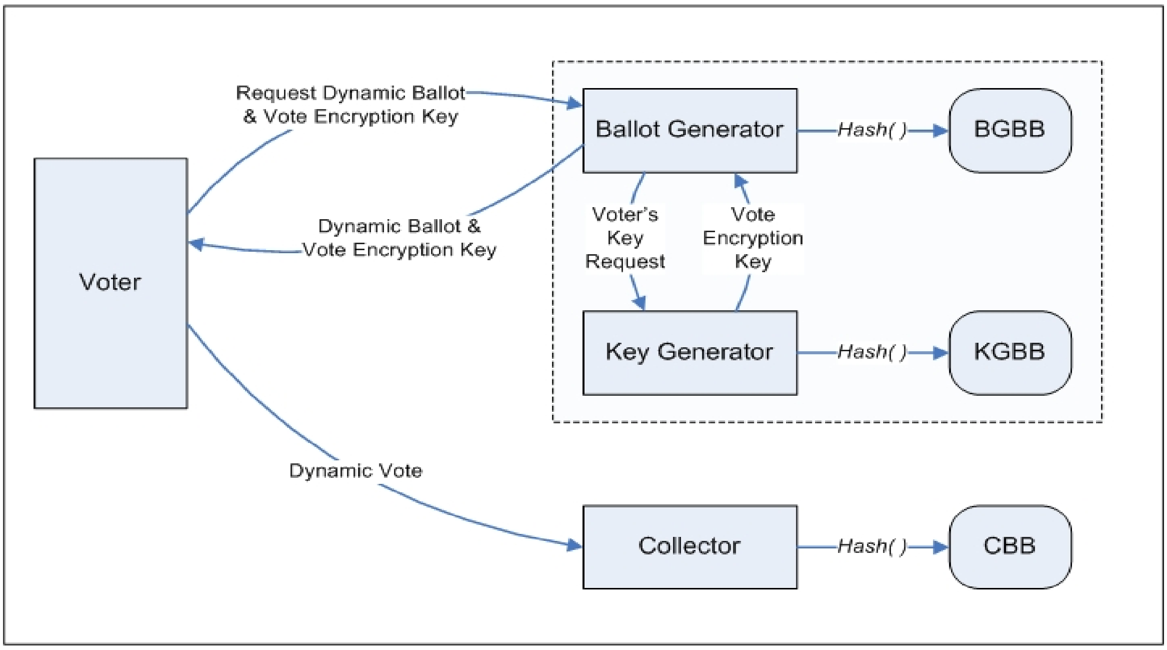


Figure 2: Overview of the voting stage

3.1.3 Counting stage

Counting stage is performed after the election period has been completed. Before proceeding with the counting of votes, Ballot Generator announces the generated ballot list; Key Generator announces the generated vote encryption key list and Collector announces the encrypted dynamic votes which include partial election information. They then send the complete lists to the Counter. Counter compares the complete lists and announced lists for consistency and checks against the hash values in the bulletin boards. Any passive observer or organization can also check the consistency of the election by using the announced lists and bulletin boards.

Counter processes the lists by decrypting the encrypted dynamic votes with the corresponding private keys and produces the dynamic vote list. Later, Counter matches the dynamic votes with corresponding dynamic ballots and obtains the actual votes. Now votes are easily tallied and the election result is announced. Lastly, the election result is verified by Key Generator.

In all stages bulletin boards are employed in order to increase security and trust in the protocol. Authorities append information to their local bulletin boards in different steps of the protocol. Thus, in each stage the voter can check and individually verify intermediate outcomes against bulletin boards. In case of any corruption he can make an objection. All communications with the bulletin board are public and therefore can be monitored. Data already written to a bulletin board cannot be altered or erased.

4. DynaVote implementation

In the previous section, we briefly described the overview of DynaVote. In this section, we will introduce the prototype of the protocol. The prototype simulates the typical voting process. The basic scenario of the protocol over Internet is as follows: (1) Voter obtains two PVIDs by using *PVID application web page* on the Election web site. (2) He accesses to the *Voting web page* on the Election web site by using $PVID_1$. He selects his candidate from the ballot list provided by Ballot Generator and casts his vote by using $PVID_2$. (3) When the election times out, *Counter application* is used to count the votes and to announce the election result.

In order to implement this scenario we have developed a client/server web application with Java. Voters represent the client side and authorities represent the server side. Servers are designed as Java applications and clients are designed as Java applets embedded in HTML files. Java applets are executed in a sandbox by web browsers, preventing them from accessing to local file system.

The voter should provide his private key while establishing a connection with PVID Authority server. Furthermore, on the voting stage he provides his PVIDs. For keeping the implementation user friendly we should not force the voter to memorize his public-private key pair and the PVIDs. Thus, prototype allows voters to save and load those data into files located in flash disk. Due to the fact that a Java applet is executed in a sandbox, we used signed applet to be able to access a local file system. This is a facility that current web browsers allow for an extension of an applet’s execution space beyond the sandbox. When a signed applet arrives on the user’s system, the user is notified of the identity of the applet’s signer and of the capabilities that applet requests. Then the user can give permission for only required capabilities.

We have used JDK 1.6 (Java 2009) for software development. Therefore, the system can be installed and executed on any computational platform with Java Virtual Machine (JVM). For the cryptographic functions, Java Cryptography Architecture (JCA) and Java Cryptography Extension (JCE) frameworks are used. In our implementation Sun JCA which includes JCE are used since it’s available with JDK 1.6. JCA consists of a set of packages and provides several cryptographic services. In this architecture, a variety of cryptographic algorithms are supported. We have used RSA public key algorithm, DES symmetric key algorithm, SHA1 PRNG (Pseudo-Random Number Generator), and SHA-256 cryptographic hash function. For database operations we have utilised the sql package of Sun. In addition to those packages, we have also used java.math.BigInteger and java.math.SecureRandom classes.

A voter connects to the servers on a TCP (Transmission Control Protocol) socket. We have used Sockets, ServerSockets, InputStream and OutputStream classes for communication between client and servers. We utilised the multi-thread support of Java in order to allow voters to connect simultaneously. For each request servers create a thread and different voters may concurrently access the server.

We have used MySQL 5.0 database (MySQL 2009) to store election data. MySQL provides an opportunity to export and import data. This opportunity is so essential to transfer data between authorities since online data transfer between authorities is not preferable. There are five databases in DynaVote prototype. BallotGenerator, KeyGenerator, Collector and Counter databases are used to store server data. BulletinBoards database is used to implement bulletin boards and it is read only accessible by all authorities and voters. As well as, each authority can write only its own bulletin board table in BulletinBoards database. The writing operation is disabled for unauthorised users.

In order to configure databases, there is an initialisation file. This file drops the existing databases and then creates all election databases with their tables required for the authorities. It initialises databases and tables.

4.1 Software packages

The software packages are explained in this section and the interaction between the packages is depicted in Figure 3.

4.1.1 e-Voting.authorities.Ballot_Generator

This package contains `BallotServerProtocol`, `BallotServer`, `BallotServerThread` and classes. `BallotServer` is the main class for Ballot Generator. The application listens on a dedicated port for voter connections and runs until the end of the election. `BallotServer` class uses multi threading. If a voter connects to the server, then an instance of `BallotServerThread` class is created.

Initially, the server backs up old election data, and then truncates the database for a new election. At the end of the election it exports data on its own database and data in BGBB (Ballot Generator Bulletin Board) table of BulletinBoards database. Hence, it creates a <BallotGenerator.sql> data file in order to send Counter authority and it announces dynamic ballots in <BallotGenerator_Result.html>.

`BallotServerThread` is a class where messages are received from the voter, checked and processed by calling `BallotServerProtocol’s` method and sent back to the voter. The thread runs until the processed messages are not equal to terminating messages. `BallotServerProtocol` class defines the communication protocol between the voter and Ballot Generator during ballot obtaining phase. Dynamic ballot is prepared in this class with SHA1 PRNG algorithm. All the transactions are written to databases.

4.1.2 e-Voting.authorities.Collector

`CollectorServerThread`, `CollectorServer` and `CollectorServerProtocol` are contained in this package. The relations between classes and general working scheme of this package are similar to `e-Voting.authorities.Ballot_Generator` package. `CollectorServer` is the main class for Collector. The application listens on a dedicated port for voter connections and runs until the end of the election. Dynamic votes are collected with associated PVIDs in `CollectorServerProtocol` class.

4.1.3 e-Voting.authorities.Counter

This package contains Counter, `CounterGUI` and `Information` classes. `Counter` is the main class for counting and tallying operations. The application provides a user interface to process all data sent by BallotGenerator, KeyGenerator and Collector and to check consistency of internal lists, published lists and bulletin boards. `Counter` class includes some methods to check whether a vote will be counted in the final tally or it will be discarded as well. `CounterGUI` class is used to provide the application user interface. `Information` class is a small class which prints the election result and gives information about the winning candidate.

4.1.4 e-Voting.authorities.Key_Generator

This package contains `KGServer`, `KGServerThread` and `KGServerProtocol` classes. In terms of the relation between classes and general working scheme, this package is very similar to the package `e-Voting.authorities.Ballot_Generator`. The application listens on a dedicated port for Ballot Generator connections and runs until the end of the election. It does not communicate with voter. Voter’s public-private voting key pair for casting his dynamic vote is generated in `KGServerProtocol` class. At the end of the election, this class produces one sql and one html file which are <KeyGenerator.sql> and <KeyGenerator_Result.html>, respectively.

4.1.5 e-Voting.database

This package contains four classes: `VoterDatabase`, `BGDatabase`, `KGDatabase` and `CollectorDatabase`. `VoterDatabase` class is implemented for voter to check consistency of data came from servers with data in BulletinBoards. Others are used for servers and they include internal lists of authorities. The classes are used to perform database operations for both authorities and voter.

4.1.6 e-Voting.PVID_Authority

This package contains `PVIDServer`, `PVIDServerThread` and `PVIDServerProtocol` classes. The relations between classes and general working scheme of this package are very similar to the package `e-Voting.authorities.Ballot_Generator`. `PVIDServer` is the main class for PVID Authority. The application listens on a dedicated port for voter connections and runs until the end of the election. `PVIDServerProtocol` class generates signed PVIDs.

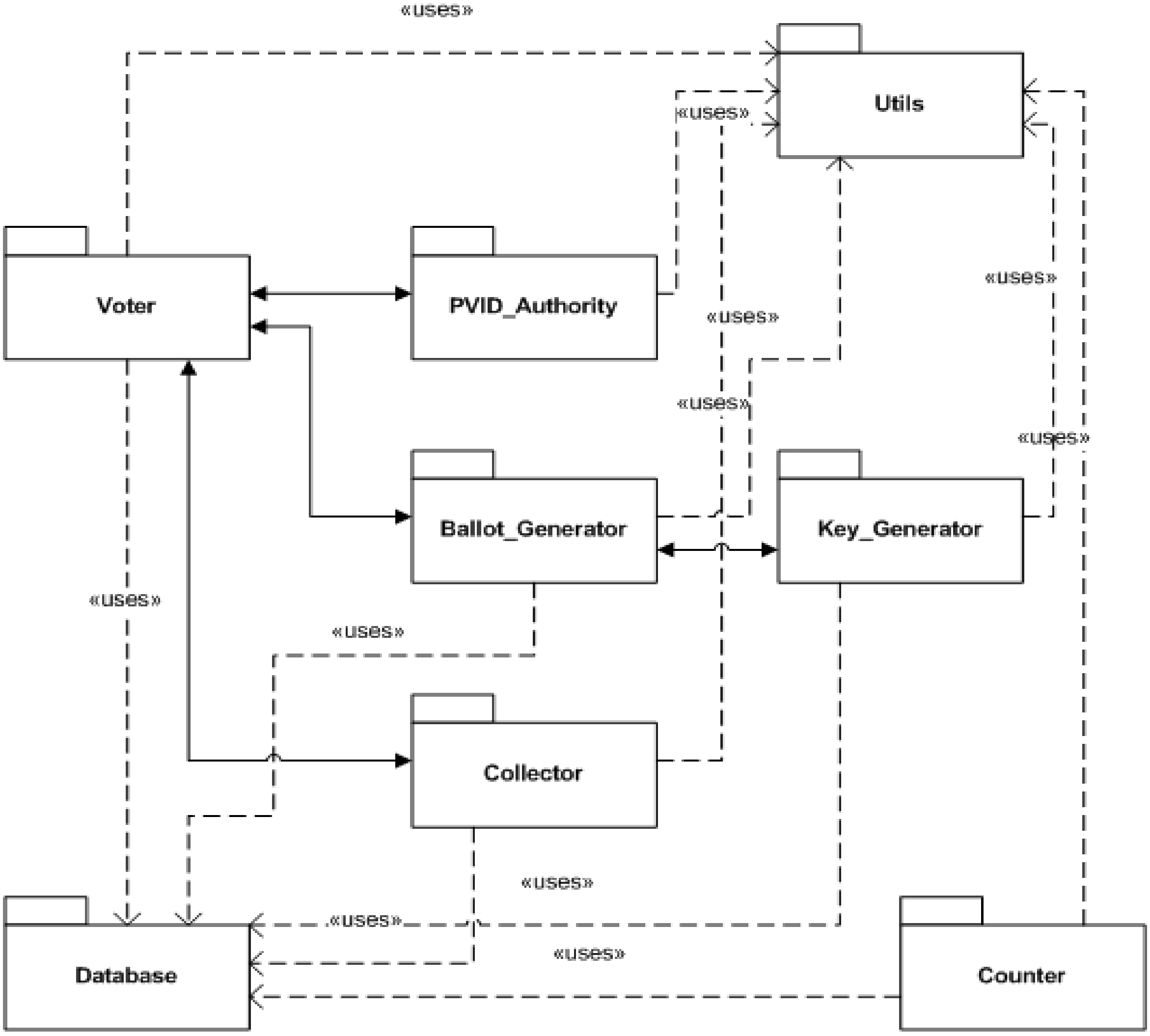


Figure 3: Package interaction and hierarchy

4.1.7 e-Voting.utils

This package contains `Constants`, `CryptoUtil`, `FileUtil`, `GUIUtil` and `Keys_Construction` classes. The methods, variables and constants in these classes are static and they are used by almost all packages.

`Constants` class includes configuration data. `FileUtil` class performs file I/O operations. `Keys_Construction` class includes methods to generate RSA public-private key pairs and to reconstruct RSA keys. `GUIUtil` class helps to produce user friendly results. It provides some conversion methods between different types and some concatenation and split operations for byte arrays. `CryptoUtil` is one of the most significant classes in the prototype, since all cryptographic functions are implemented in this class.

4.1.8 e-Voting.voter

This package contains `Voter` and `VoterGUI` classes. `Voter` is the main class for the election web application. The web page provides voters to cast their votes. Communication between voter and authorities is carried out by this class. Voter requests dynamic ballot from Ballot Generator and casts his dynamic vote to Collector. Besides, voter verifies hashed values related to his vote against KGBB

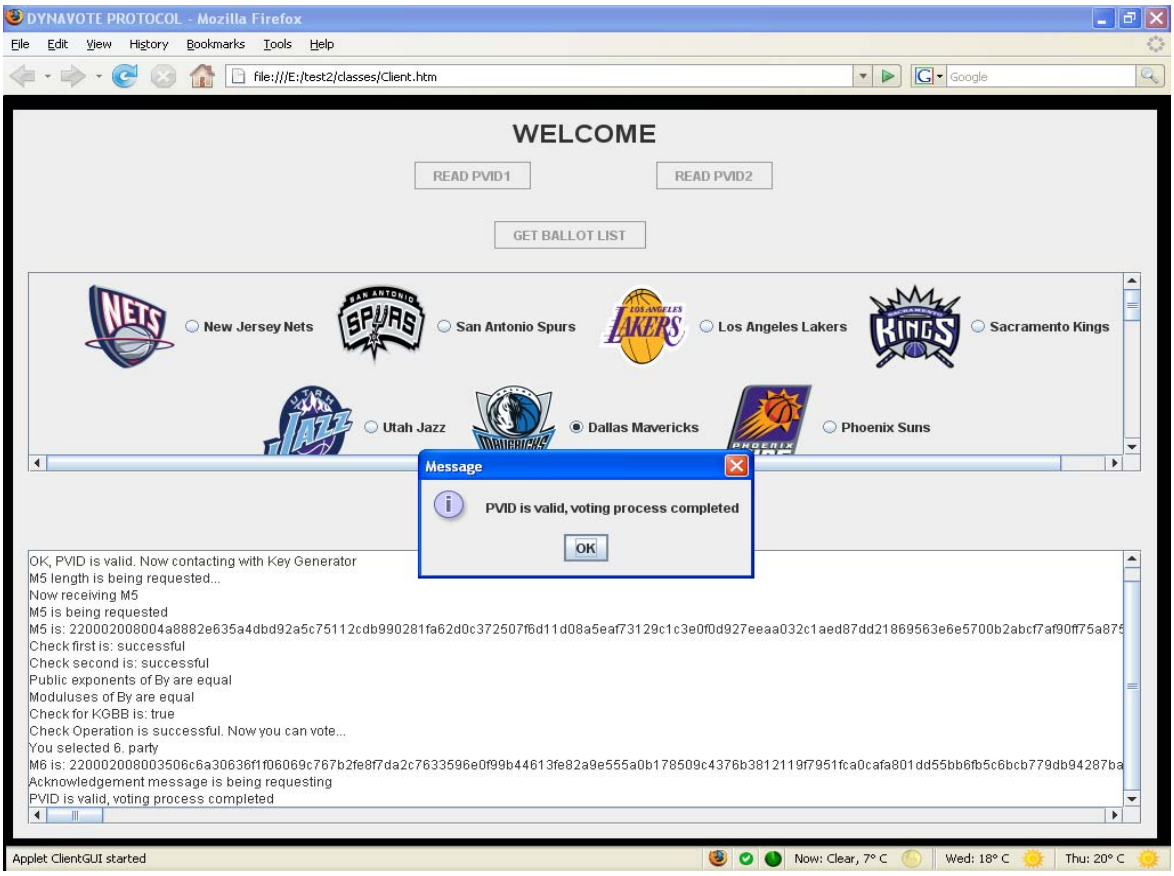
(Key Generator Bulletin Board) and BGBB databases. `VoterGUI` class is used to provide web user interface.

4.2 Prototype usage

In this section, the software usage of the prototype is described. In order to perform an election, firstly PVID Authority, Ballot Generator, Key Generator and Collector servers should be started with the same election termination time parameter on command prompt as Java applications. Afterwards any voter can access to *PVID application web page* on the Election web site over Internet to obtain signed PVIDs. *PVID application web page* has a simple user interface, which asks the voter for his registration ID and his private key. The private key is only used in client-side to encrypt voter messages.

Then any voter can access to the *Voting web page* on the Election web site and he can perform voting process if he has valid PVIDs. The Election web site uses signed applets embedded in HTML files so while using the system, voters are notified about it and the system requests permission to read the files in flash memory to be able to reach voter's private key and his PVIDs. Figure 4 shows a screen shot of the *Voting web page* which has printed after the voting process.

After election time-out, all election data in server databases are exported by authorities. These exported data are sent to Counter server in an offline way. Counter server application can be run after this point. Counter application imports all election data and then starts the counting process. During counting, it announces dynamic votes; and after tabulation, it opens a popup window that shows the winner. The number of cast votes for each candidate and their percentage are also published. Figure 5 shows a screen shot of *Counter application*.



PVID is valid, voting process completed

```
OK, PVID is valid. Now contacting with Key Generator
M5 length is being requested...
Now receiving M5
M5 is being requested
M5 is: 220002008004a8882e635a4dbd92a5c75112cdb990281fa62d0c372507f6d11d08a5eaf73129c1c3e0f0d927eeaa032c1aed87dd21869563e6e5700b2abcf7af90ff75a875
Check first is: successful
Check second is: successful
Public exponents of By are equal
Moduluses of By are equal
Check for KGBB is: true
Check Operation is successful. Now you can vote...
You selected 6. party
M6 is: 220002008003506c6a30636f1f06069c767b2fe8f7da2c7633596e0f99b44613fe82a9e555a0b178509c4376b3812119f7951fca0cafa801dd55bb6fb5c6bcb779db94287ba
Acknowledgement message is being requesting
PVID is valid, voting process completed
```

Figure 4: Voting web page

4.3 E-Voting system requirements analysis

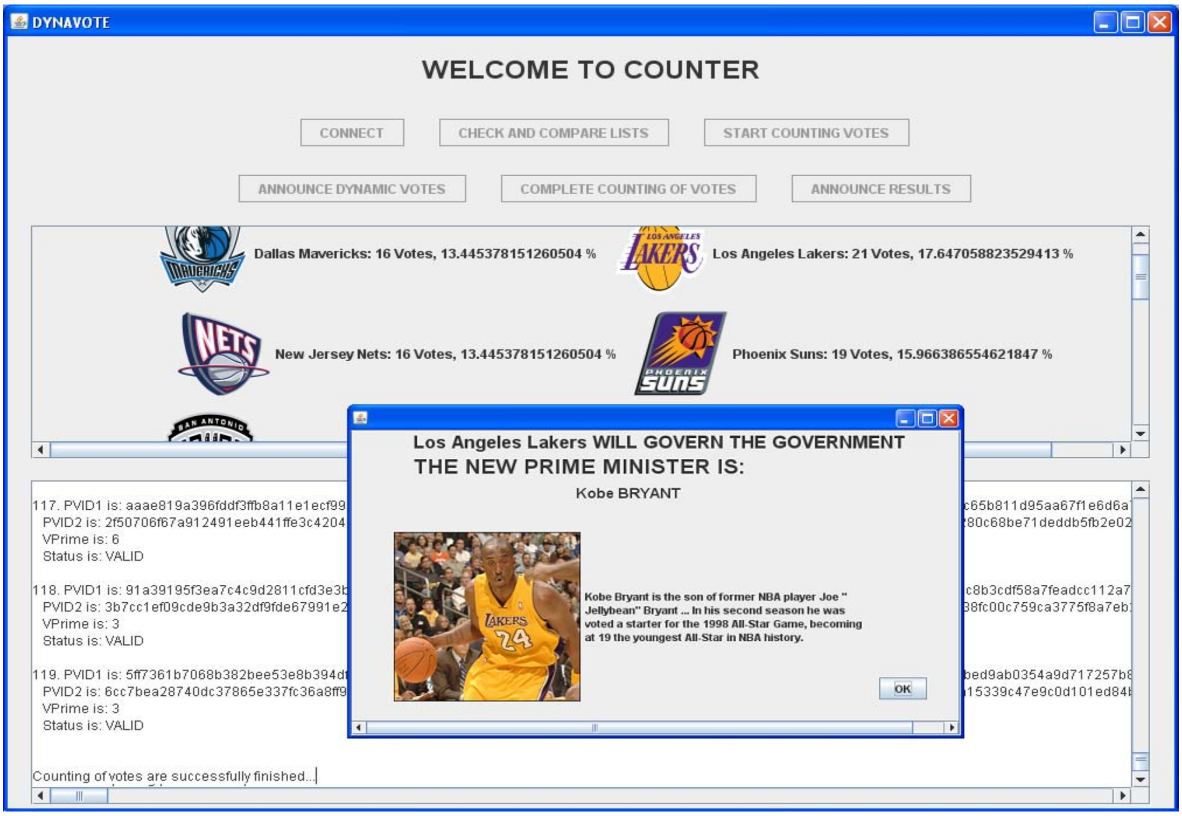
It is explained how DynaVote protocol fulfils core voting requirements (privacy, eligibility, uniqueness, uncoercibility, fairness, accuracy, robustness, individual verifiability) in (Cetinkaya 2007-2). This section analyses some e-voting system requirements and highlights how DynaVote prototype implementation satisfies them.

Efficiency (In all phases, the processes should be done efficiently): Complexity of the counting process is *O(n)* which is considerably efficient. We run some performance tests for both voting and counting processes. The detailed and comprehensive benchmark tests are described in Section 4.4.

Convenience (A convenient system allows voters to cast their votes quickly, in one session, without any extra equipment or special skills): Anyone who is familiar to use Internet can easily vote via DynaVote prototype by using its clear and easy-to-use graphical user interface. So the prototype is convenient.

Transparency (The whole voting process must be transparent): The whole voting process is transparent and bulletin boards are used to publicise the election process. The security and reliability of the system does not rely on the secrecy of the network or any other physical assumptions.

Mobility (An e-Voting system is mobile if there is no restriction on the location from which a voter can cast a vote): The prototype is mobile since the voting is performed over Internet. There is no restriction on the location from which a voter can cast his vote.



Dallas Mavericks: 16 Votes, 13.445378151260504 %

Los Angeles Lakers: 21 Votes, 17.647058823529413 %

New Jersey Nets: 16 Votes, 13.445378151260504 %

Phoenix Suns: 19 Votes, 15.966386554621847 %

Los Angeles Lakers WILL GOVERN THE GOVERNMENT

THE NEW PRIME MINISTER IS:

Kobe BRYANT

Kobe Bryant is the son of former NBA player Joe " Jellybean" Bryant ... In his second season he was voted a starter for the 1998 All-Star Game, becoming at 19 the youngest All-Star in NBA history.

```
117. PVID1 is: aaae819a396fddf3ffb8a11e1ecf99   c65b811d95aa67f1e6d6a
  PVID2 is: 2f50706f67a912491eeb441ffe3c4204   80c68be71deddb5fb2e02
  VPrime is: 6
  Status is: VALID

118. PVID1 is: 91a39195f3ea7c4c9d2811cfd3e3b   c8b3cdf58a7feadcc112a7
  PVID2 is: 3b7cc1ef09cde9b3a32df9fde67991e2   38fc00c759ca3775f8a7eb
  VPrime is: 3
  Status is: VALID

119. PVID1 is: 5ff7361b7068b382bee53e8b394d   bed9ab0354a9d717257b8
  PVID2 is: 6cc7bea28740dc37865e337fc36a8ff9   15339c47e9c0d101ed84
  VPrime is: 3
  Status is: VALID

Counting of votes are successfully finished...
```

Figure 5: Counter application

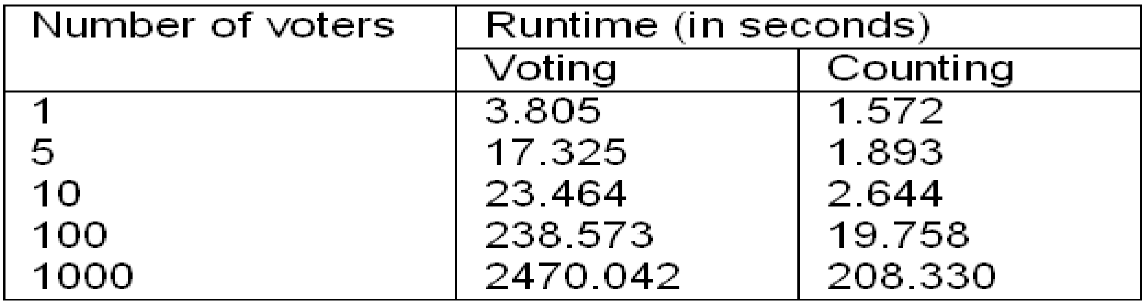
Empty Ballot (An e-Voting system should allow the voter to cast a blank vote): The prototype supports the empty ballot requirements. Those blank votes are also counted as empty ballots and cannot be invalidated, altered, deleted or copied.

Cheap Elections (The cost of the e-Voting should be less than the cost of the paper-based voting): The cost of voting by using DynaVote prototype is reasonably less than the cost of other e-Voting systems which require special hardware equipment such as DRE machines, special printers, etc.

4.4 Experimental results

In this section, we have provided performance results from our prototype for both voting and counting processes. We evaluated this performance test on Intel Pentium M processor 1.60 GHz with 752 MB RAM. For voting tests, we have simulated *n* number of voters by first creating their pseudo voter identity files. Then, for each voter, a random number *r* is created with PRNG and the party which is *r*th among all parties is cast. After the voting process is completed, votes are counted by `Counter` class. The performance results for voting and counting processes are shown in Table 1. For a sample simulation which contains 1000 voters and 1000 different votes, a result of the simulation is given in Table 2.

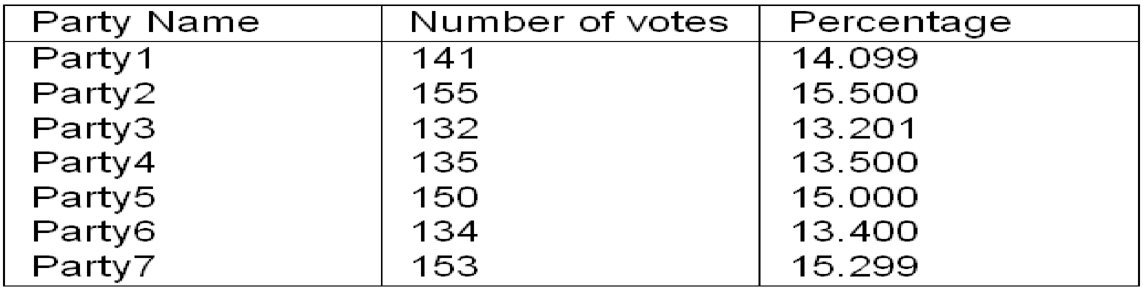
Table 1: Performance results for the prototype



Number of voters	Runtime (in seconds)	
	Voting	Counting
1	3.805	1.572
5	17.325	1.893
10	23.464	2.644
100	238.573	19.758
1000	2470.042	208.330

Performance of voting part may seem impractical at first. For testing voting, we did not use the multi-threading feature of the prototype in order to see the worst results. In other words, each vote process is performed after the previous one is completed. Since each voter has his own connection, multi-threading feature will be benefited, so practicability will be preserved.

Table 2: Election result simulation for 1000 votes



Party Name	Number of votes	Percentage
Party1	141	14.099
Party2	155	15.500
Party3	132	13.201
Party4	135	13.500
Party5	150	15.000
Party6	134	13.400
Party7	153	15.299

In the efficiency part of Section 4.3, we mentioned that complexity of counting process is *O(n)* which can also be deduced from Table 1. Considering this, it is obvious that the counting process will become less practical when the number of voters increase. However, note that simulations are performed with only one computer. Splitting the data in Counter's database into more than one part, the counting process will become easier and more practical, but while splitting, there is a very significant and delicate point. In DynaVote, a voter may cast more than one vote and the last one is counted. However, if Counter's data are split into different computers and if a voter's different casts are also divided into different computers in this splitting process, votes of same voters are counted separately. In other words, more than one vote is counted for same voters. To prevent that inconsistency, first the table has to be ordered by Pseudo-Voter Identity, and then splitting should be performed by paying attention that the same Pseudo-Voter Identities are in the same division. Otherwise, votes of the same voters are counted more than once which causes incorrect election results.

5. Conclusion

As a proof of concept, the prototype has been developed that implements the entire DynaVote protocol over Internet. The prototype includes implementation of PVID scheme component as well. This paper discusses implementation issues and improves the protocol. It presents the prototype implementation details by explaining the core design specifications and technologies used in the development; describing the packages in the prototype and providing some information about prototype usage.

This implementation study shows that improved DynaVote protocol is scalable and it is applicable to large scale elections. Since it has no physical assumption such as untappable channels, voting booths, special hardware, etc., and it has no computational complexity in all stages of the protocol. Furthermore, this prototype proves that DynaVote protocol is practical since it employs a PVID scheme which is based on blind signature. It can be performed over an uncontrolled network, such as the Internet. DynaVote has one reasonable condition, which is a recasting feature. Due to the fact that this is an acceptable high level condition related to election policy and is not a mathematical assumption; recasting can be allowed by election authorities. Thus we can conclude that DynaVote is practical in real-life.

As a future work, we will put this system into a web site in order to measure its efficiency, effectiveness and other features with real users instead of simulations. We will get feed-back from users in order to improve DynaVote. The main aim is to make DynaVote as an applicable alternative for paper based voting systems by initiating and carrying out a comprehensive project which covers every aspect of an election process. Actors other than cryptographers should participate and contribute to the project.

References

Cetinkaya O., Doganaksoy A. (2007-1) "Pseudo-Voter Identity (PVID) Scheme for E-Voting Protocols", *In Proceedings of the International Workshop on Advances in Information Security (WAIS'07) in conjunction with ARES'07*, Vienna, Austria, pp. 1190-1196.

Cetinkaya O., Doganaksoy A. (2007-2) "A Practical Verifiable E-Voting Protocol for Large Scale Elections over a Network", *In Proceedings of the 2nd International Conference on Availability, Reliability and Security (ARES'07)*, Vienna, Austria, pp. 432-442.

Cranor L., Cytron, R. (1997) "Sensus: A Security-Conscious Electronic Polling System for the Internet", *Proc. of the 30th Annual Hawaii Int. Conf. on System Sciences*, Wailea, Hawaii.

DuRette B. W. (1999) "Multiple administrators for electronic voting", *BS Thesis*, MIT.

Forsythe J. M. (2005) "Encrypted Receipts for Voter-Verified Elections Using Homomorphic Encryption", *MEng Thesis*, MIT.

Herschberg M. A. (1997) "Secure electronic voting over the World Wide Web", *MS Thesis*, MIT.

Java (2009) http://java.sun.com, last accessed 20.01.2009.

Joaquim R., Zuquete A., Ferreira P. (2003) "REVS - A robust electronic voting system", *In Proceedings of IADIS International Conference e-Society*, Portugal, pp. 95-103.

Kim K. (2002) "Killer application of PKI to Internet voting", *IWAP'02*, Springer Verlag.

Lebre R., Joaquim R., Zuquete A., Ferreira P. (2004) "Internet voting: improving resistance to malicious servers in REVS", *International Conference on Applied Computing (IADIS'2004)*, Portugal.

MySQL (2009) http://www.mysql.com, last accessed 20.01.2009.

Sampigethaya R., Poovendran R. (2006) "A framework and taxonomy for comparison of electronic voting schemes", *Elsevier Computers & Security*, Vol. 25, No. 2, pp. 137-153.

Weber S. (2006) "A Coercion-Resistant Cryptographic Voting Protocol-Evaluation and Prototype Implementatio" *Thesis*, Darmstadt University of Technology, Germany.