# A Flexible Implementation of a Web-based Election System for Educational Organizations

SHARIL TUMIN
University of Bergen
IT-Dept.
P. O. Box 7800, 5020 Bergen
NORWAY
edpst@it.uib.no

SYLVIA ENCHEVA
Stord/Haugesund University College
Faculty of Technology, Business and Maritime Sciences
Bjørnsonsg. 45, 5528 Haugesund
NORWAY
sbe@hsh.no

*Abstract:* Web-based on-line voting and on-line election systems provide benefits of usability, manageability and security. A particular workflow in any phase of on-line election process can be modeled and implemented securely by employing basic security applications readily provided by well established cryptographic technologies. By analyzing data-flow between different phases in the workflow, secure processes can be implemented using Web, database and cryptographic techniques. The implementation has to deliver a system that provides performance properties mandatory of an on-line election system of authentication, democracy, anonymity, non-coercion, accuracy, reliability, veracity, verifiability, neutrality, and likability.

*Key–Words:* e-Voting, e-Election, Applications security, Multi-tiers Web-based application, Secure Workflow Modeling, Practical Cryptographic Applications.

## 1 Introduction

The goal of this paper is to propose a Web-based election system framework for deployment within academic organizations like for example universities, collages and schools. In fact, any form of an organization with citizens having different types of roles can be benefited by employing such a voting system.

Web-based on-line voting and on-line election systems proposed in this paper will, in our view, provide three folds benefits; 1) to the organization, 2) to the voters, and 3) to the administrators. The organization is benefited by the increase in voters participation and security implementations. The voters are benefited by the usability and implemented measures that protect their privacy and rights. The administrators, either administrative or system, are benefited by providing them with a clear and secure workflow for a particular election process.

Democratic election processes are mainly participatory in nature. Citizens of a society in which a particular election is carried out need in practice to participate actively through voting in order for the society to exercise the principle of a majority rule. A democratic system, which was derived from Greek which means “popular government’, is in principle granting the power to govern to the a few elected representatives by citizens through a selection process called election.

A few details concerning who has right to vote or what a weight a vote given by a person with a particular role will depend on to which governing office a particular election is for. However, it makes no difference whether the selection process is for a vice-chancellor of a university or a president of a chess club, the election procedure will be based on the same basic rules and regulations preordained by the law of the society. Therefore, it is possible (or even desirable) to model and implement a single Web-based election system to cater for different elections in a particular society, for example an educational organization.

An essential process in representative democracies are competitive elections, that are fair both substantively and procedurally, [6]. Substantive fairness guarantees equality among all citizens in all respects and protects their rights as declared by the law of the society. Substantive fairness is a prerequisite to procedural fairness. Therefore, the clarity and exactness of the definitions that constitute substantive fairness are of a paramount importance to a successful system implementation work. Procedural fairness means that the rules of the elections are fixed in advance, easy to understand and contain no ambiguities.

The major challenges faced by the system designers and implementers are the practical interpretations of rules and regulations governing a particular society, concerning a particular election process, in another word, the “not so simple” task of mapping the substantive fairness to procedural fairness. This implies a closely working team of bureaucrats and technocrats, which by itself can be problematic. Once this work is done correctly, the rest is just a simple application of technologies.

The nature of the system to be implemented involves constraints that seem to be contradictory, for example anonymity and likability which entail privacy and at the same time a reference to a person. Anonymously linkable is a pretext to personal responsibility. In normal circumstances events are linked to anonymous references, but if necessary and this action is supported by the law any event can be linked back to a particular person with the help of some independent but related anonymous references. Whether or not anonymity means absolute privacy or implied privacy is not a discussable issue in this paper, we assume that the substantive fairness had clearly and exactly addressed this subject.

The idea of Internet voting is not new, there exist a few implementations already [2], [8], [15], and [19] since such voting schemes promise and deliver many benefits despite some difficulties to solve problems and risks [3], [5], and [14] in relation to these performance properties of authentication, democracy, anonymity, non-coercion, accuracy, reliability, veracity, verifiability, neutrality, and likability. E-voting and e-election issues are also discussed in [7], [13] and [20].

The rest of this paper is organized as follows. Section 2 contains background discussions pertinence to the issues discussed in this paper. Section 3 is devoted to the explanation of supporting tools used in the implementation of the proposed system. Section 4 explains the system itself. Finally, Section 5 contains the conclusion of this work.

## 2 Background

### 2.1 Main Actors

Simplistically speaking, an election is a process in which a set of voters make a personal selection of one person from a set of candidates for a sit in a governing office which grants the power to govern the voters in the matter related to the office for a specific period of time.

Figure 1 shows the main actors during an election process from the start to its completion.

A group of voters is a subset of all persons defined in an organization, its citizens. In a computerized system, a citizen is usually a user of the system. A user is a person with authentication and personal information defined in the domain database. More often than not, the authentications data are pairs of username and password, used by domain users to authenticate themselves during logon.

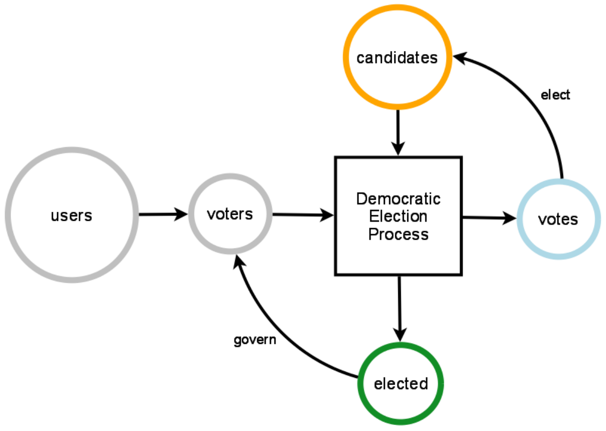


Figure 1: Main Actors in an Election Process

The number of voters depends very much on what type of office this particular election is carried out for. The whole domain users of an organization are defined as voters if the election is to elect vice-chancellor of a university. While in an election for the office of president of Stamp Collector Interest Group the number of voters maybe less than, say a hundred.

The candidates are a special group of citizens of whom the voters will choose, in the believe that she will represent the voters' best interest, if elected. Now, not all candidates will win an election. The one that received the most number of votes will be elected to the office. Therefore, unless the elected official received 100% of the voted, there is no way a governing body is representing the interest of the whole society.

The majority factor will also depend on what type of office this particular election is for. The counting procedure can be as simple as one count per vote or different votes are counted differently depending on what role voters have for a particular election process. The voters, candidates, counting procedure and majority factor have to be correctly defined and initialized before voting can take place.

Beside the fact that there are implementation details for different elections, the underlying principles and structures are the same to each and every Web-based election undertaking.

A flexible Web-based election system must capture and correctly implement these underlying principles and structures as working framework applicable to all instances of elections either large or small. The details that are needed for different type of elections can be implemented as special modules to be loaded when necessary.

One common but very important system structure is a user interface. A well designed user interface will have a positive influence on the usability scale. Since user will use the same interface for different election types, learning by experience and suggestions for improvement from users will eventually promote the system usability to optimal level. More usability means more votes, and more votes mean increase in democratic participation, which is our main goal at the first place.

## 2.2 Operational Properties

A Web-based election system must conform to specific operational standards. There are fundamental differences between a Web-based election system and a Web-based voting system.

A voting system is normally designed to measure preferences of a group of individuals to some alternatives within a problem domain. Most e-Learning systems provide some rudimentary e-Voting capabilities. While very useful in their prescribed problem domains of e-Learning environments, they are severely lacking in operational properties for supporting e-Election in terms of save guarding both substantive fairness and procedural fairness.

It is a common mistake to assume that any e-Voting implementation can be used to undertake an on-line election process. Most of the compulsory operational properties are not implemented or can not be implemented due to it underlying architectural compositions. The process of e-Election reduced to simple voting can be practically done by e-Voting, but due to fundamental laws and regulations of democratic election processes governing our political societies, it is not politically feasible. There were events concerning the anonymity issues that led to derailing of elections results in the past.

A Web-based election system needs to address practical, security and political issues. Some of these issues are contradictory to each other and will unfortunately increase the complexity of a system implementation. However complex, a system for on-line election must be user friendly. What use is an election system, no matter how good it is if it is not being used to cast votes?

As mentioned in passing in Section 1, the proposed election system must perform with the following operational properties:

**P1** - *Authentication*: Only eligible voters in a closed electoral roll shall be able to vote. A finalized *manntall* (list of eligible voters and relevant candidates) was prepared before the start of the voting period.

**P2** - *Democracy*: Each voter's votes counts only once in the election. Votes from persons with different roles may be counted differently. The rules should be clear to all voters.

**P3** - *Anonymity*: A vote shall not be associated to a voter. Privacy is a very important right a voter must have in order to ensure her personal protection.

**P4** - *Non-coercion*: A vote can not be bought or sold for any reasons. A vote can not be proved to interested parties by the voter or others after it has been cast and a choice has been made.

**P5** - *Accuracy*: All valid votes are properly counted for in the final counting. It shall not be possible to remove a valid vote from the final counting.

**P6** - *Reliability*: All erroneous, fake or otherwise non-valid votes are excluded from the final counting. It shall not be possible to include a non-valid vote in the final counting.

**P7** - *Veracity*: All voters are truthful. Each voter can cast her own vote only.

**P8** - *Verifiability*: Individual voter is able to independently verify that her vote is counted for and that the counting was done truthfully and correctly. Voters shall be able to verify that their votes have been correctly accounted for.

**P9** - *Neutrality*: The voting process must be fair both to the voters and the electoral candidates. While voting is still on the votes should be kept secret until the final phase.

**P10** - *Linkability*: Two votes from the same voter shall be linked together, but not to the person who cast them.

The property **P1** necessitates the existence of an enterprise IdM (Identity Management) system. Without an IdM an additional administration work of maintaining an authentication database for eligible voters is needed. The enterprise IdM must provide the necessary information such that non-duplication of voters in the mantall can be guaranteed. Further more, information concerning a particular persons affiliation, roles and status can be provided from the IdM to the election system.

For a particular election process, after the *mantall* is correctly assembled, "anonymified", compiled and received, an approval of the election overseers will be cryptographically signed, closed and sealed. This will directly support properties **P5** and **P6** as the *manntall* will be indirectly correlated (as needed by property **P3**) to the votes cast during the tallying phase of the election.

Properties **P8** and **P10** are concerned with similar problem of auditability. Property **P8** provides the possibility of external audits by voters, while **P10** supports internal audits. Both properties **P8** and **P10** have contradictory requirements to property **P3**.

To a greater or lesser degree a Web-based election system can be implemented to support all the operational properties mentioned here that incurred a minimal complexity increase in relation to usability of the system as a whole.

It is clear that an on-line election process needs a clear separation of the following phases; 1) initialized phase, 2) intermediate phases, and 3) finalized phase.

### 2.3 Design Principles

As democracy very much depends on citizen participation, it is a matter of great importance in our modern system of representative democracy that the majority of citizens of a society are involved in the process of electing members of governing bodies of that society. Thus a supporting system that facilitates the election process must above all be *usable*, thereafter in decreasing order of importance *trustworthy*, *fair* and *manageable*.

Therefore, in this paper we propose a Web-based election system guided by these four design principles:

**Q1** - *Usability*
**Q2** - *Trustworthiness*
**Q3** - *Fairness*
**Q4** - *Manageability*

It goes without saying that these four design principles must include and implement the ten properties mentioned earlier in Section 2.2.

A conceptual diagram of a voting process as given in Figure 1 clearly showing the cyclic nature of voting, electing and governing in a democratic election process. An elected government body has a mandate to govern for a limited period of time, until the next election.

A well designed and functioning on-line election system can be used again and again for any election purposes. To provide the flexibility needed the system will be built on a manageable system architecture.

Software components are written as modules that are easy to design, implement and maintain. Subsystems are independently managed units of functional entities that provide essential services. The system is implemented by loosely-coupled sub-systems.

By compartmentalization methodology, any components and sub-systems can be introduced, modified and customized easily and effectively.

### 2.4 Operational Constraints

Let $\mathbb{M}_{manntall}$ be the total number of voters in an electoral roll. Let $\mathbb{V}_{valid}$ be the number of valid votes counted in the result and $\mathbb{V}_{cast}$ be the total number of votes being cast during the election period. Then,

$$\mathbb{V}_{valid} \leq \mathbb{M}_{manntall} \leq \mathbb{V}_{cast}$$

Let $r \in [1, n]$ be different type of voters' roles such that,

$$\mathbb{M}_{manntall} = \sum_{r=1}^{n} \mathbb{M}_r$$

then,

$$\mathbb{V}_{valid} = \sum_{r=1}^{n} \mathbb{V}_r \text{ and } \mathbb{V}_r \leq \mathbb{M}_r, r \in [1, n]$$

Let $m$ be the number of candidates. Thus for the total sum of votes given to each candidate $s \in [1, m]$ is,

$$\mathbb{V}_{valid} = \sum_{s=1}^{m} \mathbb{V}_s$$

Let denote valid votes given by voters with a role $r$ to a candidate $s$ as $\mathbb{V}_{(r,s)}$, $r \in [1, n]$, $s \in [1, m]$. Then,

$$\mathbb{V}_{valid} = \sum_{r=1,s=1}^{(n,m)} \mathbb{V}_{(r,s)}$$

Let $\mathbb{W} = \{w_r\}$ be a set of weights assigned to each role $r \in [1, n]$, and let $\mathbb{T}_{mantall}$ be the total possible tally and $\mathbb{T}_{election}$ be the total tally of the election. Then,

$$\mathbb{T}_{mantall} = \sum_{r=1}^{n} (\mathbb{M}_r \times w_r)$$
$$\mathbb{T}_{election} \leq \mathbb{T}_{mantall}$$
$$\mathbb{T}_{election} = \sum_{s=1}^{m} \mathbb{T}_s$$
$$\mathbb{T}_s = \sum_{r=1,s}^{n} (\mathbb{V}_{(r,s)} \times w_r), s \in [1, m]$$

The winning candidate of the election is $p$ when the tally $\mathbb{T}_p$ satisfies the following condition:

$$(\mathbb{T}_p - \mathbb{T}_q)/\mathbb{T}_{election} > \delta \quad p, q \in [1, m], p \neq q$$

We have a simple majority condition when $\delta = 0$. Whenever the winning condition is not satisfied, then a new round of voting needs to be called.

### 2.5 Election Process Flow

A simplified process flow schematic is presented in Figure 2 as a use case diagram. Different human actors and their interactions with the system are clearly labeled.

From the human actors point of view, a particular system eliciting an on-line election procedure seems to have a three distinct phases of 1) registration, 2) voting, and 3) tallying. Both the electoral candidates and voters need to be registered to the system. A person's right to register as a voter in the registration phase will automatically grand her a right to cast a vote during the voting phase. A vote cast during the voting phase will be correctly counted during the tallying phase.

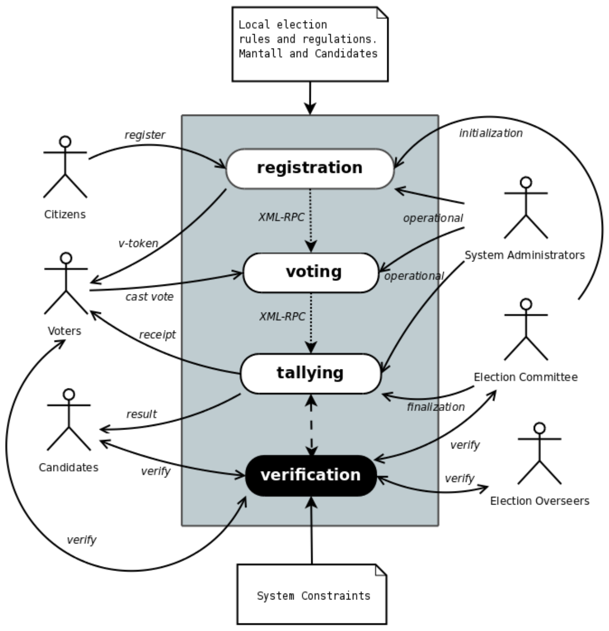


Figure 2: Election Process Flow

A voter will be given a voting token during the registration phase to be used as anonymous authentication mechanism during the voting phase. A vote is cast and counted only to those that have a valid voting token.

The system, looking from administrators and election overseers prospectives, is a bit more complicated. The system will have six distinct phases of 1) pre-election, 2) initialization, 3) registration, 4) voting, 5) tallying, and 6) post-election. These phases are executed and arranged with a strict sequential ordering as been given here constrained by prescribed durations and executive controlling actions.

In order to implement a system that satisfies the ten operational properties mentioned earlier in Section 2.2, the system is split into three administratively independent parts. System administrative responsibilities will be given to three non-collaborative teams. These teams of administrators will not share administrative secrets and sub-systems' data.

The verification sub-system is a part of tallying service where voters, candidates, election committees and election overseers can securely validate election results which preserve voters' anonymities.

## 3 Supporting Tools

### 3.1 Cryptographic Tools

The proposed system employed three well known and widely used cryptographic tools. They are 1) symmetric encryption implemented in Blowfish [18], 2) asymmetric encryption implemented in RSA [10] public-key cryptography and 3) cryptographic hash functions implemented by SHA [9] hash algorithms.

Symmetric encryption is also commonly known as a secret-key encryption. A symmetric encryption scheme, for example Blowfish, uses a shared secret key for both encryption and decryption. The strength of encryption depends on the length of the key used. Longer keys give stronger encryption.

Blowfish Encryption scheme:-
*Encryption* $\beta_{enc}$:
ciphertext, $\mathbf{c} = \beta_{enc}(K_{sec}, \mathbf{m})$
*Decryption* $\beta_{dec}$:
plaintext, $\mathbf{m} = \beta_{dec}(K_{sec}, \mathbf{c})$
*Inverse transformation*:
$\mathbf{m} = \beta_{dec}(K_{sec}, (\beta_{enc}(K_{sec}, \mathbf{m})))$

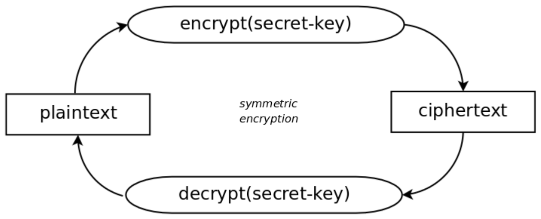


Figure 3: Symmetric Encryption

By employing a single secret-key, symmetric encryption schemes are both easier to implement and faster in execution than asymmetric encryption schemes. However there are downsides.

To maintain a secure communication channel, say between Alice and Bob, using a symmetric encryption, Alice and Bob will have to share a key, which they will keep secret. Now, if Bob wants to communicate secretly with Charlie, they then need to share a secret key, which no one else knows. If Diana also wants to secretly communicate with Bob, then she and Bob will have to share another secret key. At this point, Bob needs to know and keep secret three keys. So, in order for Bob to securely (privately) communicate with *n* number of persons, he needs to keep *n* secret keys.

In order to ensure secure communication channels between *m* persons communicating with each and everyone a total of $m(m-1)/2$ secret keys are needed. Everyone of the $m$ persons will need to securely keep $m-1$ keys.

Asymmetric encryption is also known as public-key encryption. In an asymmetric encryption a pair of mathematically related keys is used, one key from the pair is used for encryption while the other key is used for decryption. It is called public-key encryption because the key that was made public from the pair is used for encryption.

Bob creates a pair of keys to be used in asymmetric encryption. Bob keeps his private-key private and gives his public key part of the pair to Alice, Charlie and Diana. All three can now communicate with Bob securely (privately) using Bobs public-key. Neither of them can intercept and decrypt others encrypted massages to Bob, since only Bob has his private-key needed to decrypt these massages.

There are no secret keys for encryption. The only keys that need to be kept save are the private-keys.

Public-key cryptographic system depends heavily on computational complexity theory and number theory. RSA [17] is the most well known and widely used cryptographic system in today's digital world. RSA supports asymmetric-key cryptographic schemes for 1) encryption/decryption, 2) sign/verify and 3) blind/unblind.

RSA Encryption scheme:-
*Encryption* - $\xi_{enc}$
ciphertext, $\mathbf{c} = \xi_{enc}(K_{pub}, \mathbf{m})$
*Decryption* - $\xi_{dec}$
plaintext, $\mathbf{m} = \xi_{dec}(K_{prv}, \mathbf{c})$
*Inverse transformation*:
$\mathbf{m} = \xi_{dec}(K_{prv}, (\xi_{enc}(K_{pub}, \mathbf{m})))$

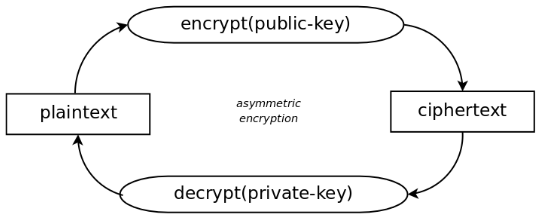


Figure 4: Asymmetric Encryption

RSA Signature scheme:-
*Signing* - $\zeta_{sig}$
signature, $\mathbf{s} = \zeta_{sig}(K_{prv}, \mathbf{m})$
*Verification* - $\zeta_{ver}$
verify, $\mathbf{v} = \zeta_{ver}(K_{pub}, \mathbf{s})$
*Inverse transformation*:
$\mathbf{m} = \zeta_{ver}(K_{pub},(\zeta_{sig}(K_{prv}, \mathbf{m})))$

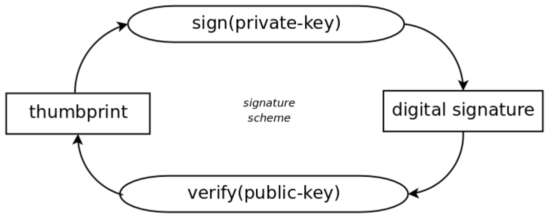


Figure 5: Signature Scheme

RSA Blind Signature scheme:-
*Blind* - $\lambda_{blind}$
blind, $\mathbf{b} = \lambda_{blind}(K_{pub}, \psi, \mathbf{m})$
*Signing* - $\zeta_{sig}$
signature, $\mathbf{bs} = \zeta_{sig}(K_{prv}, \mathbf{b})$
*Unblind* - $\lambda_{unblind}$
unblind, $\mathbf{s} = \lambda_{unblind}(K_{pub}, \psi, \mathbf{bs})$
*Verification* - $\zeta_{ver}$
verify, $\mathbf{v} = \zeta_{ver}(K_{pub}, \mathbf{s})$

The public key $K_{pub}$ is readable by all interested parties while the private key $K_{prv}$ is kept secret and only known to the owner of the key. The blinding factor $\psi$ is only known by the message owner. Blind signature is useful when anonymity is important [4]. By using blind/unblind, a signee can sign the message without knowing what it contains.

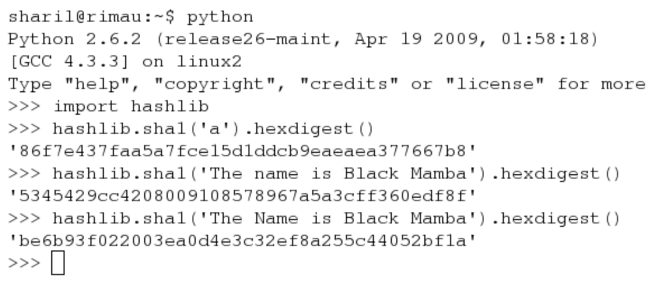
For a more detailed discussion on Public-Key Cryptography Standards (PKCS) #1 v2.1 by RSA Laboratories we refer to RFC3447 [10].

A cryptographic hash function is a procedure that takes any variable length string of characters and converts it into a fixed size hash value usually represented by a hexadecimal number. Different strings will have different hash values.

A small change in a string will produce a big difference in hash values. A hash value, sometimes called a message digest, is useful as a thumbprint of a particular string. The Python codes shown in Figure 6 will serve as proof to the points mentioned here.

By employing these ready made cryptographic tools, communication between two instances, either through space or time, can be made that supports the stringent regulatory demands for privacy, authenticity and non-reputability needed for this type of Web-based applications.

The most widely used cryptographic hash functions are MD5 [16] and SHA-1. A successful attack



```
sharil@rimau:~$ python
Python 2.6.2 (release26-maint, Apr 19 2009, 01:58:18)
[GCC 4.3.3] on linux2
Type "help", "copyright", "credits" or "license" for more
>>> import hashlib
>>> hashlib.sha1('a').hexdigest()
'86f7e437faa5a7fce15d1ddcb9eaeaea377667b8'
>>> hashlib.sha1('The name is Black Mamba').hexdigest()
'5345429cc4208009108578967a5a3cff360edf8f'
>>> hashlib.sha1('The Name is Black Mamba').hexdigest()
'be6b93f022003ea0d4e3c32ef8a255c44052bf1a'
>>> 
```

Figure 6: SHA code examples

on SHA-1 was reported in 2005. At the end of 2008, it was reported that the MD5 hash has been broken. Therefore it is advisable to use SHA-2 or SHA-3 to ensure the long-term robustness of an application that employed hash function as a part of its security implementations.

## 3.2 Software Tools

The system is implemented as a multi-tiers Web-based application, built using free and open source software. The users using their Web browsers will interact with an Apache [1] Web server using HTTPS (Hypertext Transfer Protocol Secure) communication protocol. The programmable environment and the middlewares are written in Phython [12]. The back-end database is implemented using PostgreSQL [11].

The Python programming environment is used as integrating middleware between the Web server front-end and the database back-end. Some of the important Python modules used to implement the system were:

- mod_python - Python dynamic programmable environment to Apache,
- Crypto - cryptographic libraries,
- hashlib - cryptographic hash functions,
- xmlrpclib - XML (Extensible Markup Language) RPC (Remote Procedure Call),
- tlslite - SSL v3 (Secure Sockets Layer) and TLS v1 (Transport Layer Security) libraries,
- pyPgSQL - API (application programming interface) to PostgreSQL,
- standard Python libraries for examples - SocketServer, BaseHTTPServer, base64 and binascii.

By utilizing these Python modules, different system components were constructed as independent units. These applications models can be modeled, designed, implemented and tested individually. This style of development is supportive to the development efforts within the agile operating environment of a typical Web-based application.

A system of loosely coupled sub-systems as shown in Figure 7 is normally developed in many implement-test iterations. These prototype-driven development cycles are said to be completed when all concerned parties are satisfied with the current prototype. The development efforts will convene at a later time whenever new issues arise beyond the capabilities of the maintenance team, for example fixing a designed flaw or implementing new functionalities.

## 4 System

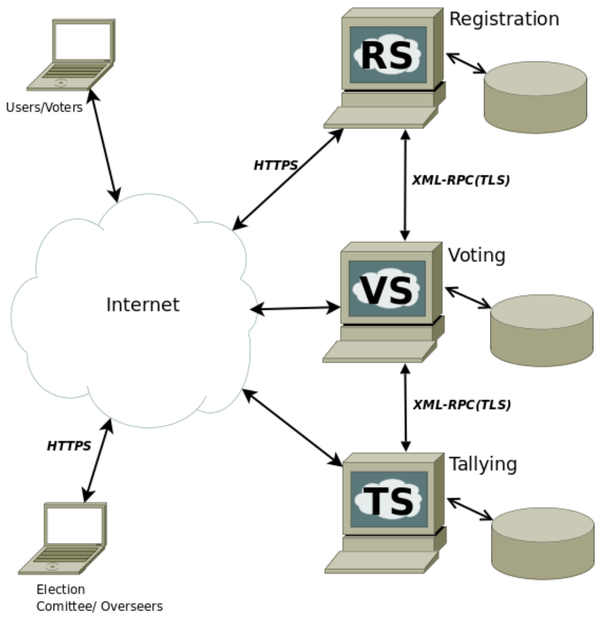


Figure 7: System Architecture

### 4.1 System Architecture

The proposed system is based on multi-servers, distributed authorities and multi-tiers application architecture as shown in Figure 7. There are a minimum of three separate servers controlled by three different administrative authorities. These servers implement all supporting functions using multi-tiers Web-based application techniques where presentation, business logic and datastore are implemented as separate but connected layers. The three servers are loosely connected to each other using XML-RPC (XML-based remote procedure call) over TLS.

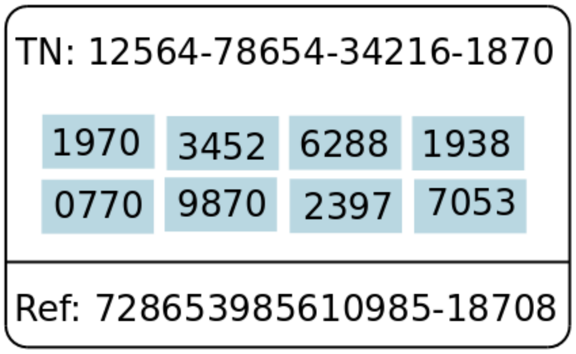
These three servers will facilitate the system by performing different tasks for the different stages and functions of the election process flow introduced in Section 2.5. These servers, in relation to their main functions, are labeled as follows:

**RS** - *Registration server*
**VS** - *Voting server*
**TS** - *Tallying server*

Their utilization will follow the path of a typical election process; 1) voters registration, 2) voting, and 3) votes counting.

## 4.2 Operational Phases

The finish product must support the **Operational Properties** of Section 2.2. The system must be designed to operate securely, correctly and responsively in accordance to the **Design Principles** of Section 2.3.



TN: 12564-78654-34216-1870

| | | | |
|---|---|---|---|
| 1970 | 3452 | 6288 | 1938 |
| 0770 | 9870 | 2397 | 7053 |

Ref: 728653985610985-18708

Figure 8: Voting Token Card

The task of designing, installing and operating an e-Election system is both challenging and complex. The complexity maybe related to the multiplicity of purpose and contradictory issues such as 1) anonymity/accountability, 2) privacy/auditability, and 3) secrecy/verifiability.

It is conducive to these operational demands that the operational phases should be modeled out in advance in order to be 1) documented, 2) agreed upon and 3) followed as guidelines by all participating parties. A group of election overseers will monitor the whole election process (whether manually, semi-automatically or automatically) as it progresses through its phases:

(i) ***pre-election*** - Define the election ground rules such that the operational constraints of Section 2.4 can be implemented unambiguously. Prepare *manntall* ,i.e a list of eligible voters and actual candidates. It is important that the *manntall* must be closed and sealed before the voting get started, i.e it is not possible to trick the system to count votes from bogus voters.

Let $\Pi$ and $\Omega$ are election overseers. Let Blowfish secret key $B_\Pi$, public key $P_\Pi$, and private key $Q_\Pi$ belong to $\Pi$. Let Blowfish secret key $B_\Omega$, public key $P_\Omega$, and private key $Q_\Omega$ belong to $\Omega$.

Each voter in the *manntall* will be represented by a data triplet {VID, role, EID}. Voter identification 'VID' is a unique large random number, 'role' is the role the voter in the organization (e.g. student, lecture, professor, engineer), and 'EID' is Blowfish encoded username, $EID = \xi_{enc}(P_\Pi, usename)$. Each candidate will also be represented similarly with CID serve as candidate identification, $CID = \xi_{enc}(P_\Omega, usename)$.

The data triplets are written, one line for each voter, to the *manntall* file $F_{m0}$. This file is then encrypted using Blowfish to produce $F_{m1}$, $F_{m1} = \beta_{enc}(B_\Omega, F_{m0})$. The hash value $H_m$ is calculated by applying the SHA256 hash function on $F_{m0}$. The hash value $H_m$ together with a timestamp $TS$ is then signed using $Q_\Pi$, $S_m = \zeta_{sig}(Q_\Pi, (H_m + TS))$. The Blowfish secret key $B_\Omega$ is encrypted using public-key encryption to produce $PB_\Omega$, $PB_\Omega = \xi_{enc}(P_\Omega, B_\Omega)$.

The produced items $F_{m1}$, $S_m$, $TS$, and $PB_\Omega$ are given to the overseer $\Omega$ for save keeping and to be used during tallying phase in the election process.

(ii) ***initialize*** - All voters listed in the *manntall* must be defined in the enterprise IDM (identity management system) which will provide the initial authentication during voters registration phase. Different servers (**RS**, **VS**, **TS**) will identify voters using different referring methods. These methods of identification will protect each voter anonymity and at the same time provide authentication at different phases of the election process.

**RS** - ID(PWD) $\Longrightarrow$ VID, where ID (authenticated by password PWD) is a username and VID is a number, uniquely given to each voter.

**VS** - VID $\Longrightarrow$ TN(PIN), where TN (authenticate by PIN) is a voting token card number, an example is shown in Figure 8.

**TS** - TN $\Longrightarrow$ *Ref* a voting token card reference number, *Ref* $\Longrightarrow$ ballot.

Database at **RS** must be initialized with voters ID, VID values, where VID must correspond to the value in the *manntall* file $F_{m0}$.

(iii) ***registration*** - In order to vote, users need to register to the **RS**. Users authenticate themselves by their ID and PWD. An authenticated user becomes a voter

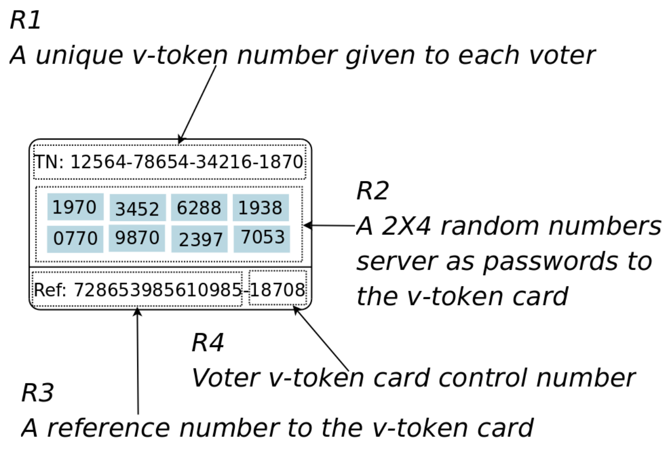

R1
A unique v-token number given to each voter
TN: 12564-78654-34216-1870
1970 3452 6288 1938
0770 9870 2397 7053
R2
A 2X4 random numbers server as passwords to the v-token card
Ref: 728653985610985-18708
R4
Voter v-token card control number
R3
A reference number to the v-token card

Figure 9: Voting Token Card Usage

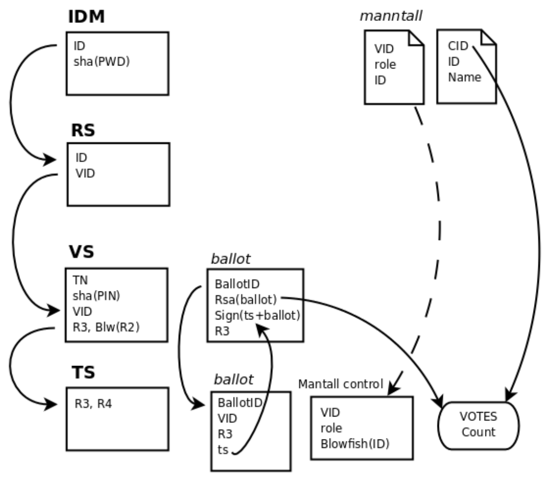


Figure 10: Data Reference Diagram

when 1) the user is in the *manntall* and 2) a dynamically created voting token card is issued by the **VS**.

There are different regions on the token card for different purposes as shown in Figure 9. The voting token passwords *R2* are encrypted by Blowfish encryption using a user provided PIN as secret key, $R2_{enc} = \beta_{enc}(PIN, R2)$. Note that the PIN is known only to the voter. The SHA256 hash value $H_{PIN}$ of the PIN, the encrypted token passwords $R2_{enc}$, *R1* and *R3* are stored in **VS**. These data will be used to authenticate voter later in the voting phase.

The token card reference number *R3* and the control number *R4* are sent over to **TS** and are stored there as shown in Figure 10. The *Registration server* **RS** serves only as an authenticator to eligible voters when obtaining voting token cards.

The administration of voting tokens is the responsibility of the *Voting server* **VS**. A voter can, via **RS** re-issue a voting token, if for example the voter has forgotten her PIN. The **VS** will re-issue the card with the same *R1*, *R2*, *R3* and *R4* but with different $R2_{enc}$ and $H_{PIN}$ if a different PIN is used. Self-service tokens re-issuing mechanism is also useful when cards are stolen or misplaced.

(iv) ***voting*** - Using a voting token card, a voter can cast her vote. She can do this as many times as she wants as long as the voting period is still on. However, only the latest ballot of a particular voter with a multiple cast ballots will be counted by the *Tallying server* **TS**. The possibilities of re-casting of votes will discourage the practice of vote-selling. A seller can not give a conclusive proof to a buyer of the sold vote that the vote will be included in the final count. After all the seller can sale the vote token to another buyer or she can always re-cast another vote at any time herself.

A particular voter will use *R1* and PIN during the initial authentication process. Let $R1_{in}$ and $PIN_{in}$ are values given by the voter at the authentication checkpoint. The voter is authenticated when $R1_{in} = R1$ and the SHA256 hash value of $PIN_{in}$ is equal to $H_{PIN}$.

The $PIN_{in}$ will be used to decrypt $R2_{enc}$, $R2 = \beta_{dec}(PIN_{in}, R2_{enc})$. The **VS** will then present to the authenticated voter an empty ballot to be used for voting. Also, on this ballot Web-form are input fields for three randomly chosen passwords $R2_{(3of8)}$ to be filled by the voter as passwords, which will serve to authenticate the ballot.

The voter will know which of the three out of the eight, four digits passwords, to be used by their positions on the 2×4 matrix *R2*, for example; $R2_{(1,3)}$, $R2_{(2,2)}$, and $R2_{(1,2)}$, and in that order. These positions will be presented to the voter by using a simple visualization graphic. So, with this example, $R2_{(3of8)} =< R2_{(1,3)}, R2_{(2,2)}, R2_{(1,2)} >$.

On this ballot the voter will mark her vote or votes, depending on election, and provide the correct combination of three *R2* passwords. The ballot is validated by **VS** by checking the correctness of the provided $R2_{(3of8)}$ against the stored $R2$ in terms of both values and sequence ordering. The correct $R2_{(3of8)}$, as given in the example, for a particular voting card as shown in Figure 8, will be *6288 9870 3452*.

The cast vote *V* in the authenticated ballot will be encrypted by **VS** using the RSA public key of Ω to produce $V_\Omega$, $V_\Omega = \xi_{enc}(P_\Omega, V)$ and at the same time a new unique *BallotID* is calculated. The $V_\Omega$ indexed by the *BallotID* is saved into the database.

The **VS** sends a message to **TS** containing

{*BallotID*, VID, R3}. The **TS** will save this item plus a timestamp $ts$ in its database and will reply by sending the $ts$ back to **VS**.

The **VS** will then blind the $V$ plus the $ts$ to produce $b_{V+ts}$, $b_{V+ts} = \lambda_{blind}(P_{Omega}, \psi, (V+ts))$. The **VS** will send the $b_{V+ts}$ to **TS**. The **TS** will then blindly sign the $b_{V+ts}$ to $bs_{V+ts}$, $bs_{V+ts} = \zeta_{sig}(Q_{Omega}, bs_{V+ts})$. As a reply, **TS** will send the calculated $bs_{V+ts}$ back to **VS**. The **VS** will then unblind $bs_{V+ts}$ to produce $s_{V+ts}$, $s_{V+ts} = \lambda_{unblind}(P_{Omega}, \psi, bs_{V+ts})$. And finally the **VS** stores the produced $s_{V+ts}$ into the database referred by *BallotID*.

The voter is then redirected to **TS** together with a GET parameter *BallotID*. At this point, the **TS** will present the voter with a receipt together with *R3* to prove its authenticity. The **TS** will ask the voter to provide *R4* to be used as a proof that the voter received the receipt.

If the provided *R4* is equivalent to the stored *R4* then the **TS** will then mark the ballot referred by *BallotID* as valid and will notify the voter of this fact. The voter is given the opportunity of three retries for providing the **TS** with a correct *R4*. The ballot will be marked as invalid after three wrong attempts.

This marks the end of a voting process of a particular voter.

The encrypted ballots containing the votes and their signatures are stored in **VS**. System administrators or any person with similar access rights at **VS** can not know the contents of the ballots since they do not have the private key to decrypt the ballots. Since *R4* for any particular vote token is not stored in **VS**, persons with enough access at **VS** can not submit a valid ballot even though both VID and *R3* are known to them.

Similarly, system administrator at **TS** cannot know the particular of a ballot since **TS** only stores references to the ballots saved at **VS**. The signing application at **TS** signed the ballots blindly. Person with enough access at **TS** can not store a bogus ballot at **VS** because they have no access there. Storing a bogus ballot reference even with an existing *BallotID* will not constitute a security breach because such references will fail the verification process of a particular $s_{V+ts}$.

(v) ***tallying*** - At a predefined date, the Web servers at both **RS** and **VS** will be stopped. This marks the end of the voting period. However, the counting of votes does not commence immediately. There is an intervening period when voters can verify that theirs votes are registered at **TS**.

Anyone can consult the **TS** with a *R3* and the result of this inquiry is a response from **TS** containing an acknowledgment that a voter having a vote token with that particular *R3* had cast her vote and that her ballot was registered in the system. This message will be given regardless to the number of time the voting token was used or by whom.

During this period the trusted election overseers will collect all relevant log files from all three servers and save them in a safe off-line media. These log files will be used during investigations should such cases become necessary to resolve future conflicts and uncertainties.

The **TS** will open the *manntall* file $F_{m0}$ and reads each VID and its associated role. The $PB_{\Omega}$ is decrypted using the private key $Q_{\Omega}$ to get the Blowfish secret key $B_{\Omega} = \xi_{enc}(Q_{\Omega}, PB_{\Omega})$. The file $F_{m1}$ is then decrypted using Blowfish to produce $F_{m0}$, $F_{m0} = \beta_{dec}(B_{\Omega}, F_{m1})$. The hash value $H_m$ is calculated by applying the SHA256 hash function on $F_{m0}$. The hash $H_m$ together with a timestamp $TS$ is then used to verify the signature $S_m$ using $P_{\Pi}$, $V_m = \zeta_{ver}(P_{\Pi}, S_m)$. If $(H_m + TS)$ is equal to $V_m$, then the signature is valid and the content of file $F_{m0}$ has not been altered.

Valid voting data $\{V_{\Omega}, s_{V+ts}\}$ of the latest cast ballot of every *R3* are read from **VS** database and stored at **TS**. These ballots are related by the equality of their ballot reference numbers *BallotID*. The $V_{\Omega}$ is decrypted to get $V$, $V = \xi_{enc}(Q_{\Omega}, V_{\Omega})$. The $V$ together with $ts$ is then used to verify $s_{V+ts}$ using verification function $\zeta_{ver}(P_{\Pi}, s_{V+ts})$.

The VIDs will be used to control the votes validity and the roles will be used as weighting factors during counting. The counting of votes can start after the **TS** finishes decrypting, verifying the signature and VID-validity checking of each ballot now stored in its database. The counting will follow the rules defined in Section 2.4.

The overseers will finally announce the election winners after a manual validation of the results are done.

(vi) ***post-election*** - Administrators will take database backup and do clean-up on all three servers. They also delete all pertinence log files in relation to security and voters privacy on all three servers. The backup media must be stored securely. The privacy of voters must be protected even when the election was over. The database backup and the saved log files should be destroyed after a period of time according to prescribed regulations.

## 5 Conclusion

A good security is a matter of implementing simple and understandable secure procedures within a workflow rather then implementing complicated security protocols and using difficult cryptographic tools.

We have discussed the complexity of on-line election systems due to their contradictory requirements. We have come up with a solution based on distribution of authorities with a limited administrative power over each other and by limiting the amount of unwanted collaboration between them.

The security is modeled based on operations done on the servers and the interaction between them. The security implementation employs simple and well known cryptographic techniques.

The responsibilities of managing and upholding operational security are given to trusted election overseers and are not placed in the hands of a system administrator.

We preserve anonymity and at the same time promote verifiability and likability by using three voting stages of registration, voting, and tallying.

By letting the voter the possibility of re-issuing tokens and re-casting of votes the system discourages the practice of vote-selling.

The proposed system gave a flexible framework for Web-based election system built on the framework of a group of loosely coupled sub-systems. Different election procedures, either large or small, can be accommodated by the system by simply re-programming the tallying module without affecting other system modules.

The clients (Web browsers) need only to support HTML, Web-cookies and page redirect. The clients do not need to run Java or JavaScript. We believe running programs on clients reduces security and usability of s system.

We have discussed the design and implementation of a Web-based election system scalable up to 10000 voters. This number is based on our experience managing an IDM with 10000 active users. It is possible to accommodate a higher number of voters by dividing the voters in groups and each group will be served by different **VS** and **TS** installations and a master **TS** will then collect all valid ballots from the subordinate **TS**s.

*References:*

[1] Apache, The Apache Software Foundation, HTTP server. *http://www.apache.org/* 2009 (last accessed)

[2] BigPulse Online Voting. *http://www.bigpulse.com/elections* 2009 (last accessed)

[3] Benaloh, J., Tuinstra, D.: Receipt-free secret-ballot elections (extended abstract). *STOC 94: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pp. 544553 ACM Press, 1994 pp. 544553

[4] Chaum David, Blind signatures for untraceable payments, *Advances in Cryptology - Crypto '82*, Springer-Verlag (1983), 199-203

[5] Cranor L.F., Cytron R.K.,Design and implementation of a practical security- conscious electronic polling system *Technical Report WUCS-96-02, Washington University* (1996)

[6] Democracy, From Wikipedia, the free encyclopedia. *http://en.wikipedia.org/wiki/Democracy* 2009 (last accessed)

[7] Drigas, A. S., Koukianakis, L., E-Government Application for Supporting a Network od Distributed Public Administration Units, WSEAS Transactions on Computers, 3(6), (2004), pp. 33-36

[8] Helios Voting Elections you can audit. *http://www.heliosvoting.org/* 2009 (last accessed)

[9] FIB PUB 180-3, SECURE HASH STANDARD (SHS). *Federal Information Processing Standards Publication*, 2008

[10] J. Jonsson, B. Kaliski, Public-Key Cryptography Standards (PKCS) #1. *RSA Cryptography Specifications Version 2.1 RSA Laboratories* , 2003

[11] PostgreSQL, The world's most advanced open source database. *http://www.postgresql.org/* 2009 (last accessed)

[12] Python Programming Language. *http://www.python.org/* 2009 (last accessed)

[13] Penjira (Mony) Kanthawongs, An Analysis of the Information Needs For E-Parliament Systems, WSEAS Transactions on Information Science and Applications, Vol.1(5), (2004), pp. 12-37

[14] Piles J. J, Salazar J. S., Ruiz J.,Moreno-Jimenez J. M., *Security Considerations in e-Cognocracy* A. Levi et al. (Eds.): ISCIS 2006, LNCS 4263, pp. 735-744 Springer-Verlag Berlin Heidelberg (2006)

[15] Punchscan See your vote count. *http://punchscan.org/* 2009 (last accessed)

[16] R. Rivest, The MD5 Message-Digest Algorithm. *MIT Laboratory for Computer Science and RSA Data Security, Inc*, 1992

[17] R. Rivest, A. Shamir & L. Adleman, A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21 (2), 1978, pp. 120-126

[18] B. Schneier, The Blowfish Encryption Algorithm.
*http://www.schneier.com/blowfish.html*
2009 (last accessed)

[19] Votehere Proves every vote was counted properly. *http://votehere.com/old/default.php*
2009 (last accessed)

[20] Yannas P. and Lappas G., E-Campaign in Greek Elections: 2000-2004, *WSEAS Trans. on INFORMATION SCIENCE and APPLICATIONS*, 5(1), (2004), pp. 1332-1338