Web-based Election System for Small Scale to Medium Scale Academic Societies

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: On-line voting and on-line election workflow can be done securely by employing basic security applications readily provided by well established cryptographic technologies. By analyzing data-flow between actors in the workflow secure processes can be implemented using Web, database and cryptographic techniques. The implementation must deliver a system that provides performance properties of authentication, democracy, anonymity, no coercion, accuracy, reliability, veracity, verifiability, neutrality, and linkability.


1 Introduction

The goal of this paper is to propose a Web-based election system framework for deployment within academic organizations 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.

Democratic processes are mainly participatory in nature. Citizens of a society need to participate actively through voting in order for the society to exercise the principle of a majority rule. Democracy, which was derived from Greek, after all means “popular government”.

An essential process in representative democracies are competitive elections, that are fair both substantively and procedurally [6]. Substantively fairness guarantees equality among all citizens in all respects and protect their rights as declared by the rules of the society. Procedural fairness means that the rules of the elections are fixed in advance, easy to understand and contain no ambiguities.

The idea of Internet voting is not new, there exist a few implementations already [2], [7], [13], and [17] since such voting schemes promise and deliver many benefits despite some difficulties to solve problems and risks [3], [5], and [12] in relation to these performance properties of authentication, democracy, anonymity, no coercion, accuracy, reliability, veracity, verifiability, neutrality, and linkability.

2 Background

2.1 Operational Properties

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

P1 - Authentication: Only eligible voters in a closed electoral roll shall be able to vote. A finalized mantall (list of eligible voters) was prepared before the start of 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 - No coercion: A vote can not be bought or sold for any reasons. A vote can not be proved 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.
2.2 Design Principles

As democracy very much depends on citizen participation, it is a matter of great important in our modern system of representative democracy that majority of citizens of a society involved in the process of electing members of governing bodies of that society. Thus the supporting system that facilitate 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 2.1.

2.3 Operational Constraints

Let $M_{mantonall}$ be the total number of voters in the electoral roll. Let $V_{valid}$ be the number of valid votes counted in the result and $V_{cast}$ be the total number of votes being cast during the election period. Then,

$$V_{valid} \leq M_{mantonall} \leq V_{cast}$$

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

$$M_{mantonall} = \sum_{r=1}^{n} M_r$$

then,

$$V_{valid} = \sum_{r=1}^{n} V_r$$

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

$$V_{valid} = \sum_{s=1}^{m} V_s$$

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

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

Let $\mathcal{W} = \{w_r\}$ be a set of weights defined to each role $r \in [1, n]$, and let $T_{mantonall}$ be total possible tally and $T_{election}$ be total tally of the election. Then,

$$T_{mantonall} = \sum_{r=1}^{n} (M_r \times w_r)$$

$$T_{election} \leq T_{mantonall}$$

$$T_{election} = \sum_{s=1}^{m} T_s$$

$$T_s = \sum_{r=1}^{n} (V_{(r,s)} \times w_r), s \in [1, m]$$

The winning candidate $p$ of the election is when the tally $T_p$ satisfies the following condition,

$$(T_p - T_q) / T_{election} > \delta \quad p, q \in [1, m], p \neq q$$

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

2.4 Election Process Flow

A simplified process flow schematic is presented in Figure 1 as a simple use case diagram. Different human actors and theirs interactions with the system are clearly labeled.
Proceedings of the 9th WSEAS International Conference on DISTANCE LEARNING and WEB ENGINEERING

The three main election phases of 1) registration, 2) voting and 3) tallying, are designed to be operational and administrative independent of each other. System administrative responsibilities will be given to three non-collaborative teams. These administrators will not share administrative secrets and 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 simple, well known and widely used cryptographic tools. They are 1) symmetric-key encryption implemented in Blowfish [16], 2) public-key encryption implemented in RSA [9] public-key cryptography and 3) cryptographic hash functions implemented by SHA [8] hash algorithms.

Blowfish symmetric-key encryption uses a shared secret key for both encryption and decryption.

Blowfish Encryption scheme:-

- Encryption $\beta_{enc}$:
  
  $c = \beta_{enc}(K_{sec}, m)$

- Decryption $\beta_{dec}$:
  
  $m = \beta_{dec}(K_{sec}, c)$

- Inverse transformation:
  
  $m = \beta_{dec}(K_{sec}, (\beta_{enc}(K_{sec}, m)))$

RSA Signature scheme:-

- Signing - $\xi_{sig}$
  
  $s = \xi_{sig}(K_{prv}, m)$

- Verification - $\xi_{ver}$
  
  $v = \xi_{ver}(K_{pub}, s)$

RSA Blind Signature scheme:-

- Blind - $\lambda_{blind}$
  
  $b = \lambda_{blind}(K_{pub}, \psi, m)$

- Signing - $\xi_{sig}$
  
  $bs = \xi_{sig}(K_{prv}, b)$

- Unblind - $\lambda_{unblind}$
  
  $s = \lambda_{unblind}(K_{pub}, \psi, bs)$

- Verification - $\xi_{ver}$
  
  $v = \xi_{ver}(K_{pub}, s)$

Public-key cryptographic system depends heavily on computational complexity theory and number theory. RSA [15] is the most well known and widely used cryptographic system in today’s digital world. RSA supports non-symmetric-key cryptographic schemes for 1) encryption/decryption, 2) sign/verify and 3) blind/unblind.

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]. Using blind/unblind, a signee can sign the message without knowing what it contains.

RSA Encryption scheme:-

- Encryption $\xi_{enc}$
  
  $c = \xi_{enc}(K_{pub}, m)$

- Decryption $\xi_{dec}$
  
  $m = \xi_{dec}(K_{prv}, c)$

- Inverse transformation:
  
  $m = \xi_{dec}(K_{prv}, (\xi_{enc}(K_{pub}, m)))$

Please refer to RFC3447 [9] for a more detail discussion on Public-Key Cryptography Standards (PKCS) #1 v2.1 by RSA Laboratories.

The most widely used cryptographic hash functions are MD5 [14] and SHA-1. A successful attack 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 Python [11]. The back-end database is implemented using PostgreSQL [10].

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 packages used to implement the system were 1) mod_python - live-programmable module to Apache, 2) Crypto - cryptographic libraries, 3) xmlrpclib - XML (Extensible Markup Language) RPC (Remote Procedure Call), 4) tlsllite - SSL v3 (Secure Sockets Layer) and TLS v1 (Transport Layer Security) libraries, 5) pyPgSQL - API (application programming interface) to PostgreSQL and 6) standard Python libraries for examples - SocketServer, BaseHTTPServer, base64 and binascii.
4 System

4.1 System Architecture

The proposed system is based on multi-servers, distributed authorities and multi-tiers application architecture as shown in Figure 2. 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 2.4:

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

![Figure 2: System Architecture](image)

4.2 Operational Phases

The finish product must support the Operational Properties of Section 2.1. The system must be designed to operate securely, correctly and responsively in accordance to the Design Principles of Section 2.2.

![Figure 3: Voting Token Card](image)

The task of designing, installing and operating an e-Election system is both challenging and complex. The complexity may relate 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 2.3 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 voting is started, i.e. it is not possible to trick the system to count votes from bogus voters.

(ii) **initialize** - Each voter in the *manntall* must be defined in the enterprise IDM (identity management system). Different servers (RS, VS, TS) will refer to voters using different references such that voters anonymity are preserved and at the same time provide authentication.

- **RS** - ID(PWD) ⇒ VID, where ID (authenticate by password PWD) is username and VID is a large random number unique to each voter.
- **VS** - VID ⇒ TN(PIN), where TN (authenticate by PIN) is voting token card number, an example is shown in Figure 3.
- **TS** - TN ⇒ Ref voting token card reference number, Ref ⇒ ballot.
A unique v-token number given to each voter

A 2X4 random numbers server as passwords to the v-token card

Voter v-token card control number

A reference number to the v-token card

Figure 4: Voting Token Card Usage

(iii) registration - In order to vote, users need to register to the RS. Users authenticate themselves by their ID and PWD. An authenticate user becomes a voter when 1) the user is in the manntall and 2) a dynamically voting token card is issued by the VS.

There are different regions on the token card for different purposes as shown in Figure 4. The voting token passwords R2 are encrypted by Blowfish encryption using a PIN as secret key. Note that the PIN is known only to the voter. The SHA256 hash value of the PIN, the encrypted token passwords, 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. The Registration server RS serves only as 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, R3 and R4 but with different R2. 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 time 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 both re-issuing of token and re-casting of votes will discourage the practice of vote-selling. Since a vote saler can not give a conclusive proof of her choice to a vote buyer beyond simple trust between them.

The voter will use R1 and PIN during the initial authentication process. The PIN will be used to decrypt R2. The VS will then present to the authenticated voter an empty ballot to be use for voting. Also, on this ballot Web-form are input fields for three randomly chosen positions from R2 to be filled by the voter as passwords, which will serve to authenticate the ballot.

The authenticated ballot will be encrypted by VS using the RSA public key of TS. At the same time the VS blinds the ballot and sends it over to TS to be signed blindly by TS using RSA private key of TS. The encrypted ballot, ballot signature, timestamp, R3 and unique index number for this particular voting are saved at VS.

The voter is then redirect to TS together with parameters containing the voter’s VID, timestamp, R3 and the voting index number. Finally, the TS will present the voter with a receipt together with R3 to prove of it authenticity. The TS will ask the voter to provide R4 to be used as proof that the voter received the receipt. The TS will then save the vote cast as given in the parameters.

(v) tallying - At a predefined date, the Web servers at both BS and VS will be stopped. This mark the end of 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 respond from TS containing the receipt and the submitted R4 of the latest ballot related to the R3.

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 investigation should the case arises.

The TS opens the manntall and reads each VID and its associated role. The VIDs will be used to control the votes validity and the roles will be used as weighing factors during counting.

Voting data related to the latest ballot of each R3 is read from VS and stored at TS. The counting of votes can start after the TS finishes decrypting, verifying the signature and VID-validity checking of each ballot. The counting will follow the rules defined in Section 2.3 and finally announce the election winners.

(vi) post-election - Administrators take database backup and do clean-up on all three servers. They also delete all pertinence log files 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 within a system framework. We have discussed the complexity of on-line voting system due to contradictory requirements. We have come with a solution based on distribution of authorities with limited administrative power over each other. The security is modeled on the servers and the interaction between them only. We employed simple and well known cryptographic techniques.

We preserve anonymity and at the same time promote verifiability and linkability by using two stage voting processes of registration and voting. By letting the voter the possibility re-issuing tokens and recasting of votes the system discourage the practice of vote-selling.

The clients (Web browsers) need only to support HTML and page redirect. The clients do not need to run Java or JavaScript. We believe running program on clients reduce security and usability of the 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 installation and a master TS will then collect all valid ballots from the subordinate TSs.

References:


