e-Voting Requirements and Implementation

Rachid Anane[1], Richard Freeland[2] and Georgios Theodoropoulos[2]
[1]*Computer and Network Systems, Coventry University, UK.*
r.anane@coventry.ac.uk
[2]*School of Computer Science, University of Birmingham, UK.*
{r.freeland, g.k.theodoropoulos}@cs.bham.ac.uk

Abstract

The level of research that e-voting has attracted is a testimony of its importance as a key element in the implementation of e-government. It is argued that the ease with which voting can be performed will increase participation and enhance accountability. This convenience however, generates a set of specific requirements, not least the ability of the underlying distributed system to model the behaviour of manual systems. More specifically, the elimination of direct physical intervention entails a careful management of the implications of virtual participation. The scope of this work concerns the identification and integration of specific mechanisms for addressing issues of security, privacy and accountability. The aim of this paper is to present a case study on the design and implementation of an e-voting prototype system, and to provide a context for the selection and deployment of relevant mechanisms.

1. Introduction

With the ubiquity of the Internet, e-voting is often presented as the ultimate manifestation of the democratic process, especially when applied to the conduct of general elections. The ease with which the voting process can be carried out electronically has the potential to foster greater public participation and to enhance accountability. The adoption of electronic voting in local and national elections is often justified by the benefits it is expected to offer [1, 2]:

Mobility: e-voting would allow voters to cast their vote from a convenient location instead of being required to travel to a specific polling station.

Increased participation: electronic voting has the potential to maximise turnout at elections. An Internet based system might appeal to younger voters who are currently less likely to exercise their vote.

Increased efficiency and accuracy: e-voting promises to remove some of the human errors inherent to the manual systems especially in the counting of votes. A computerised system it is argued is more efficient and allows for a quicker delivery of results.

Transparency: e-voting is also expected to be more open to public scrutiny. However this demand for openness should not compromise security.

With the exception of Estonia in the 2007 elections, most countries have so far resisted calls for a full conversion to systems that would allow voters to vote electronically from unrestricted remote locations. Reports on the evaluation of the deployment of such schemes have pointed to a number of prohibiting factors, mostly related to security and voter authentication [3]. Moreover, the debate over the Diebold system and its vulnerabilities has also highlighted the importance of interfacing and the requirements for an acceptable level of security in e-voting schemes [4]. The lack of maturity of deployed systems has led many to argue that closer scrutiny of the source code by independent experts should be a requirement, and that open source software in this respect offers more scope for transparency and verifiability [5].

The concern over the deployment of e-voting systems stems primarily from the fact that while a distributed system removes the need for direct physical intervention it must however address the implications of virtual participation. In addition to the support for the democratic process, as modelled by the manual system, a distributed system must ensure the integrity of the voting process itself, and manage adequately the emerging behaviour of the underlying mechanisms. These concerns have driven the research in this area with a particular emphasis on enhancing the voting process. The identification of specific requirements has led to the design and refinement of e-voting models and to the implementation of various mechanisms [6]. The scope of this work falls under the latter category. The aim of this paper is to present a case study on the design and implementation of an e-voting prototype system and to provide a context for the selection and deployment of relevant mechanisms.

The remainder of the paper is organised as follows. Section 2 details the requirements for e-voting schemes. Section 3 gives an outline of some design considerations and Section 4 presents the architecture of the prototype system. Section 5 deals with system interaction and viability. Section 6 highlights some fundamental issues for further investigation and Section 7 presents some conclusions.

2. e-Voting requirements

The manual management of the voting process in democratic elections follows a specific pattern made up of four tasks:

- The registration is concerned with the setting up of the list of eligible voters (Administrator).
- The validation entails checking the eligibility of potential voters and ensuring that only legitimate users can cast their vote (Validator).
- The collection involves gathering the ballots (Collector).
- The tallying consists in counting the votes (Counter).

The voting process is however performed within specific constraints that an electronically based voting system should conform to. In particular, the system must meet the theoretical requirements for a secure electronic voting scheme [7]:

- Only eligible voters are able to vote.
- No voter is permitted to vote more than once.
- No one should be able to determine the value of anyone else's vote.
- No one can duplicate a vote.
- No one can alter another person's vote without being detected.
- Voters can verify that their vote has been counted.

2.1 Core properties of the voting system

These e-voting requirements are often translated into explicit and detailed properties that a viable e-voting system should possess [8]:

Accuracy: a system is considered accurate if it is not possible to alter a vote, discount a validated vote from the final tally or include an invalid vote in the final count.

Privacy (un-traceability): this is ensured by a system that prevents any agency from linking a specific voter with the ballot he cast, and does not allow voters to prove the way they voted.

Individual and universal verifiability: individual verifiability refers to the fact that any individual can verify that his vote was received properly while universal verifiability allows a voter to verify that all votes have been counted properly. Verifiability contributes to the public trust in e-voting systems.

Eligibility: the system must ensure that only eligible voters can vote and that each voter can cast only one vote.

Fairness: early results should not be disclosed so that voters are not be influenced by intermediate results.

2.2 Underlying system properties

In addition to the inherent properties of the democratic process, there are systemic properties that are necessary for the viability of electronic voting:

Soundness or robustness: the system should ensure that the election process is not affected by illegal behaviour or faulty procedures.

Mobility: voters should be able to cast their votes from anywhere without geographical constraints. The system should also be available and accessible during the polling phase.

Integrity: system integrity ensures that a computer system must be tamper proof, and data integrity ensures that data has not been modified during transit. Integrity refers also to collusion resistance where electoral agencies are unable to conspire to introduce, modify or remove votes in an election.

Convenience: a system should allow users to cast their vote easily, quickly and with minimal instruction. Convenience is translated into usability and reliability requirements. The challenge however is to balance convenience and security.

3. Design considerations

Design issues are concerned with the identification of appropriate models and mechanisms for facilitating the voting process. The deployment of an e-voting system is inevitably associated with a distributed system, which introduces specific security and privacy issues. This requires the selection of a specific voting protocol and the design, implementation and integration of relevant mechanisms. The following are critical to the conformance to e-voting requirements:

- Secure channels to ensure privacy and secrecy.
- Anonymous channels to prevent linking a specific voter to the vote that was cast.
- A clear separation of concerns and the discharge of functions by reliable and trustworthy agencies.
- Convenient and secure interfacing.

3.1 Secure communication channels

In e-voting a communication channel is secure if it ensures secrecy through the use of symmetric or asymmetric key encryption; and ensures data integrity by means of digital signatures, message Digests or message authentication codes (MACs**).**

3.2 Anonymity

Anonymity is usually interpreted as a complete dissociation between a vote and the voter who cast it. Three main schemes were devised in support of this requirement. These schemes have formed the basis of many voting protocols:

- Schemes that make use of homomorphic encryption reduce a ballot to a number and ensure that all the voter choices are kept secret [9]. This has the advantage that the vote can be performed without

decrypting any of the ballots, but this can be computationally expensive.

- Schemes that generate mixes (mix-nets) are used to permute different entities to hide the correspondence between input and output items, and ensure that an item is only processed once [10]. In protocols that implement this scheme, the connection between voters and ballots is removed.
- Schemes based on blind signatures [6, 11] allow an agency to sign a message without knowing its contents. Schemes based on blind signatures present a number of advantages over the other two schemes. They are more flexible and can accommodate various ballot formats. Moreover, their relatively small communication and computational complexity makes them suitable for large-scale elections.

3.3 Anonymous channels

In the TCP/IP protocol suite an Internet packet carries the IP addresses of the source and the destination machines. This information is particularly important in reliable communication especially at the transport level. The TCP protocol establishes a reliable channel between processes running on different machines, and therefore between client and server. In e-voting systems the awareness by the destination machine of the IP address of the source machine, may compromise the anonymity requirement. Even if a client sends a vote without any identifying information the identity of the voter can be extracted from the IP address. Voting protocols seek to overcome this problem by implementing an anonymous channel whereby a server can reliably and securely receive messages but cannot determine the identity of the sending machine.

3.4 Separation of concerns

Separation of concerns entails allocating specific tasks to specific agencies in order to pre-empt collusion. The voting protocol proposed by Fujioko *et al* [6], often referred to as FOO, offers provable security, guarantees privacy, prevents fraud by some parties and ensures fairness. Thanks to its flexibility, efficiency and conformance to e-voting criteria, the FOO protocol has formed the basis for many voting protocols and has been the subject of various enhancements. It also offers a high degree of compatibility with manual systems [8, 12, 13]. The FOO protocol seeks to prevent collusion between Validator and Counter by removing any functional overlap between the two agencies. The protocol makes use of the blind signature scheme and works as follows: *A* is the Validator/Authenticator and *C* is the Counter:

Stage 1:

1. *Voter* completes the ballot and blinds it.
2. *Voter* constructs a message containing the ballot and his identity and encrypts it with *A*'s public key.
3. *Voter* sends the message to *A*.
4. *A* decrypts the message, validates *Voter* and signs the ballot.
5. *A* returns the blinded ballot to *Voter*.

Stage 2:

6. *Voter* unblinds the ballot and encrypts it with *C*'s public key.
7. *Voter* forwards the ballot to *C* over an anonymous channel.
8. *C* checks for *A*'s signature on the ballot, decrypts it and increments the corresponding count.

The use of a blind signature ensures the privacy of the ballot during the validation stage. There is no opportunity for *A* and *C* to collaborate during that first stage since it is the voter who is in control of the blinding algorithm. On forwarding the ballot to *C* the voter's privacy is guaranteed by using an anonymous channel. Moreover a randomisation of the ballots by the Collector can hinder time analysis by the Counter or by other rogue server. This concern over potential server collusion has also been addressed by distributing trust over many servers [14].

3.5 Interfacing

Some researchers consider that the most important issue in e-voting is interfacing [15]. Avoiding bias in voting procedures has also become one of the prime issues in e-voting implementation. As borne out by the confusion in the 2000 US elections, the underlying technology and the design of ballots can have a significant effect on the outcome of elections. This emphasis on interface also confirms the fact that the viability of e-voting depends to a large extent on the manner with which client machines are configured, eligibility is validated and accountability enforced. System designers therefore strive to strike a balance between usability and verifiability on one hand, and security on the other [16]. Some protocols rely on the use of smart cards to ensure privacy and convenience [17]. These schemes however require specific hardware which may be compromised.

4. System architecture

The proposed prototype system is based on the FOO protocol and enforces a clear separation between the different agencies. The voting protocol is implemented by the architecture shown in Figure 1. The voting system consists of five election servers responsible for conducting the voting process, and one client, which represents the voter. The system was implemented in Java, and the client server architecture was supported by servlets.

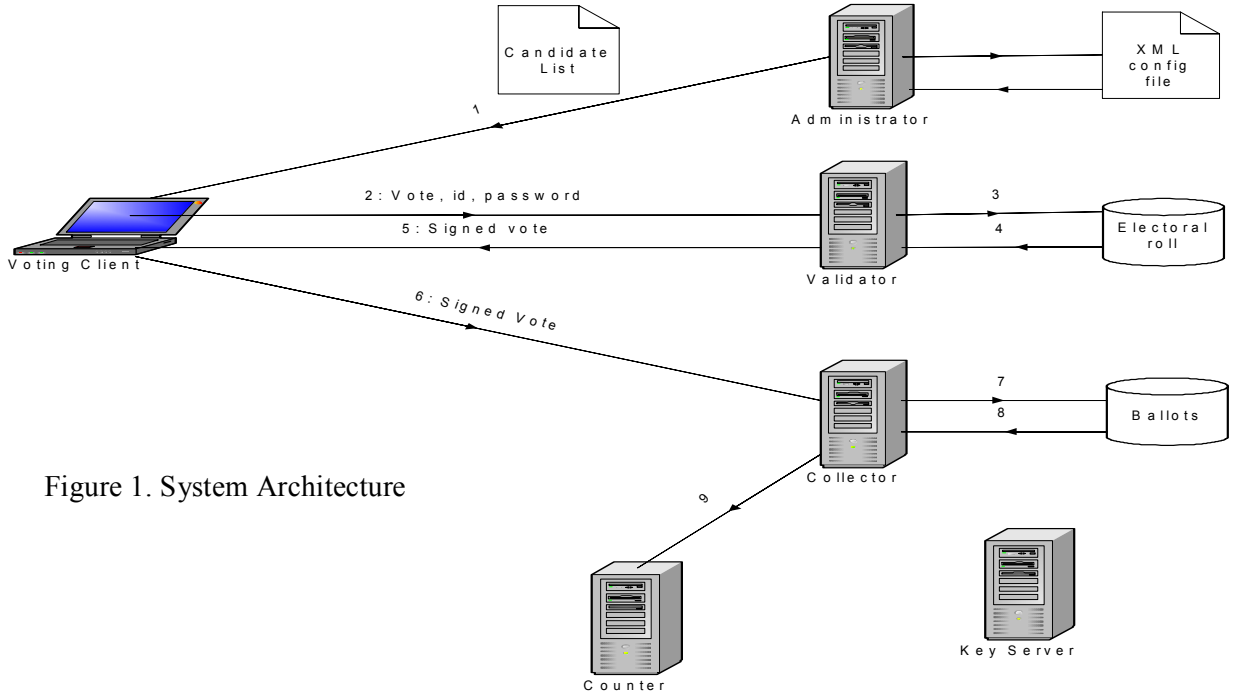

Figure 1. System Architecture

4.1 Client

A client program is required to perform a number of functions on behalf of the voter. These functions include exchanging messages with the servers, processing user input and performing the necessary cryptographic transformations. The client was implemented by an applet for security and convenience, since there is no requirement for the voter to install an application. The client performs the same functions as 'the pollster' in some systems [12].

4.2 Servers

Two servers were designed to fulfil the roles of Validator and Counter. Two additional servers were also introduced, an Administrator for distributing the list of candidates and a Collector for implementing an anonymous channel. Finally a Key Server was added for collecting and distributing the public keys to the servers. The functions of the servers are described below:

Administrator: 1.Retrieves the candidate list, 2.Sends to each voter a copy of the Validator's public key (used for the encryption of validation requests), 3.Sends to each voter a copy of the candidate list from which the voter makes his choice.

Validator: 1.Generates a public/private key pair, 2.Receives validation requests from voters, 3.Checks that the voter is eligible and that the voting code is correct, 4.If code is valid, signs the vote and returns it to the voter.

Collector: 1.Receives encrypted votes from voters, 2.Strips off any identifying information (communication headers) and stores the votes in their encrypted format until the election is complete. 3.Shuffles the votes (to prevent time analysis) and forwards them to the Counter.

Counter: 1.Generates a public/private key pair, 2.Requests the votes from the Collector, 3.Decrypts the votes, 4. Verifies that the Validator has signed each vote, 5.Tallies the votes, 6.Publishes the results to an HTML page, 7.Publishes the list of serial numbers and corresponding vote values are sent to an HTML page.

Key Server: 1.Collects public keys from the Validator and Counter, 2. Distributes the keys to the servers.

4.3 Anonymous channel

The authors of the FOO protocol left unspecified the implementation of anonymous channels. In the proposed architecture an additional server, the Collector, is introduced as an intermediary between the Voter and the Counter. The role of the Collector is to anonymise the votes by stripping off any communication headers that would allow the Counter to determine the source of the votes. The Collector also shuffles the votes before forwarding them to the Counter so as to prevent it from engaging in some form of time analysis. The use of a server as an anonymiser introduces a new process, which may be vulnerable to malicious attacks. In isolation the Collector is however powerless; even though it can obtain the identity of the voter from the IP address it cannot access the value of the vote because it is encrypted with the Counter's key. If however, the Collector can persuade the Counter to reveal its private key, the anonymity of the votes can easily be compromised.

4.4 The communication layer

The HTTP protocol is used as the basis of a first level of communication between the client applet and the servers. The fact that the connections are not persistent – they allow for only one request-response exchange before the connection is closed – is not an issue as the applet never engages in more than one exchange with any single server. An additional layer, the Secure Object Stream, was developed as an implementation of the cryptographic protocol. The client and servers communicate by exchanging serialized objects. When an object is transmitted over the HTTP connection it is first passed through the stream where the appropriate encryption scheme is applied. A proxy pattern was used to hide the complexity of the communication from the client and server classes.

4.5 Ballot life cycle

The blinding and signing operations are internal to the ballot object. The object passes through a number of distinct states between the moment it is initialised and the final count. The transition between these states is illustrated in the following steps:

1. The ballot begins in plaintext format *[m]*.
2. A random number sequence *x* is added to *m* to act as a serial number and prevent duplication of votes *[(m+x)]*.
3. *(m + x)* is concealed by raising it to the power of a random blinding factor *r, $[(m+x)^r]$,* and sent.
4. The Validator, checks voter eligibility and applies a signature *S* to the blinded ballot, using its private key *$[S((m+x)^r)]$*.
5. The voter unblinds both signature and ballot *[S(m+x)]*.
6. The Counter verifies the signature, using the public key of the Validator, and removes the random sequence to reveal the plaintext ballot *[m]*.

The transformations applied to the ballot are performed by a combination of symmetric key encryption (DES), and asymmetric key encryption (RSA). This combination allows for the data to be sent securely in both directions, and supports the signing and verification of the ballot.

4.6 Storage

The candidate list is stored in an XML file from which candidate objects are extracted using the JDOM API. As the data is not persistent and limited in volume a lightweight XML file was chosen rather than a heavyweight database. Relational databases are however used for storing the electoral roll and the submitted ballots.

Key Storage: the RSA key pairs, generated at start up, are committed to secondary storage. Should the Validator or Counter crash before the election is completed the Counter will be unable to verify the Validator's signature or decrypt the ballots. Each key pair is therefore written to the key store file as a set of numerical components, which can be used to initialise the servers. For an RSA key pair, the components are the modulus, the public exponent and the private exponent.

Secured data: some of the data stored by the system, such as the signed votes kept by the Collector, arrives in an encrypted format and is secured. Steps are also taken to secure user validation details, which, if leaked, could be used to successfully cast bogus votes. When input, the secure voting codes (passwords) are therefore transformed using a one-way hash function (SHA-1) and stored in the electoral roll database. When the voter is validated the same hash function is applied to the supplied voting code and the result is compared to the code stored in the database.

Data recovery: all votes received and stored by the Collector are backed up to a database. If a fault occurs in the primary data source, all votes saved up to that point could be recovered. To avoid the loss of the details of voters a similar approach could be applied to the electoral roll database.

5. System interaction

The integrity of the voting process can be affected by unsafe and unsecure interfaces where the voter's intent may not be recorded accurately. Convenience, therefore acquires a special significance since "the ultimate goal of secure electronic voting is to replace physical voting booths" [18]. Moreover, adequate interfacing in auditing and monitoring plays a crucial role in building and maintaining trust in the servers.

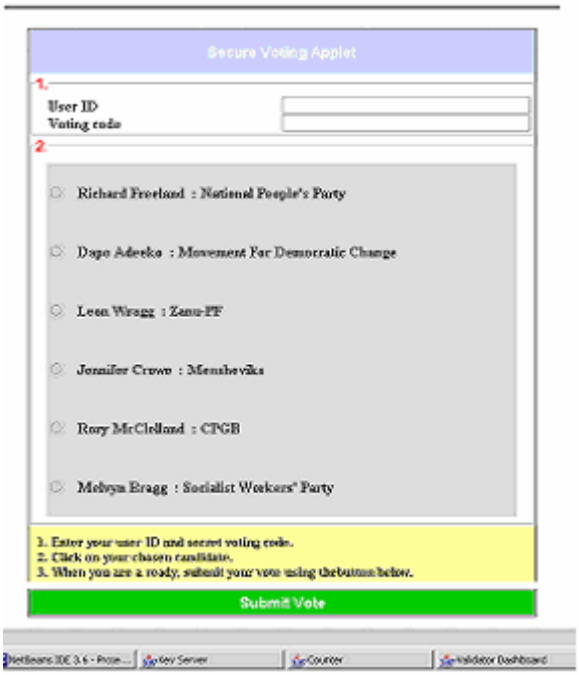

Figure 2 Voting interface

5.1 User interface

Simplicity and clarity were the overriding factors in the design of the user interface. The design was facilitated by the simplicity of the underlying two-stage protocol. All the steps required for voting are presented in one screen, as shown in Figure 2. The steps ensure an orderly progression from one stage to the next. The voter is asked to 1. Enter authentication details 2. Select a candidate and 3. Submit the vote. If the input is invalid the applet alerts the voter and returns the interaction to the initial stage. After submission the value of the vote is displayed in a message area and the voter is asked to confirm the selection. This ensures that at least two commitments are required from the voter – if the first one is a mistake it can be reversed. If, however, the voter submits and confirms a vote, which is not intended, there is mechanism for reversing the decision and the vote has effectively been recorded.

5.2 Verification

When a vote is submitted for counting, a random serial number generated by the client is attached to the ballot. The serial number is made visible to the voter via the applet. When the election result is announced, the Counter publishes the list of serial numbers, along with a list of ballot values attached to each serial number. This arrangement allows the voter to privately check that his vote was recorded accurately. Although this scheme introduces an association between a voter identity and a serial number, only the voter should be aware of this association. The privacy of the vote is therefore maintained. The Collector however has the potential to breach anonymity by storing the IP addresses of voters along with the serial numbers received from those voters. The Collector can later use the same system of verification to check which IP address voted for which candidate, assuming that machines are not shared by voters. To guard against this, the serial number (as the ballot itself) is encrypted with the Counter's public key before it is sent to the Collector.

The system does not provide any means for the voter to challenge the published results. Currently it is not possible for that voter to prove conclusively that his vote was counted wrongly. The voter could save the serial number displayed immediately after the vote is cast and prove that the value of the vote has changed. However, it is difficult to protect such a receipt against forgery and therefore the legitimacy of any challenge would still be doubtful. Moreover, the introduction of receipts may compromise privacy and may undermine legitimacy.

5.3 Monitoring

Each server presents a 'dashboard' designed to keep the administrator informed of the number of requests being handled and the status of each request (Figure 3). As the servers process the requests, information is dispatched to the relevant dashboard and displayed. An administrator should be able to gauge the level of activity of the servers. It should also be possible to monitor client machines and identify those that repeatedly send unsuccessful or fraudulent requests. The administrator might for example decide to deny access to the rogue machines.

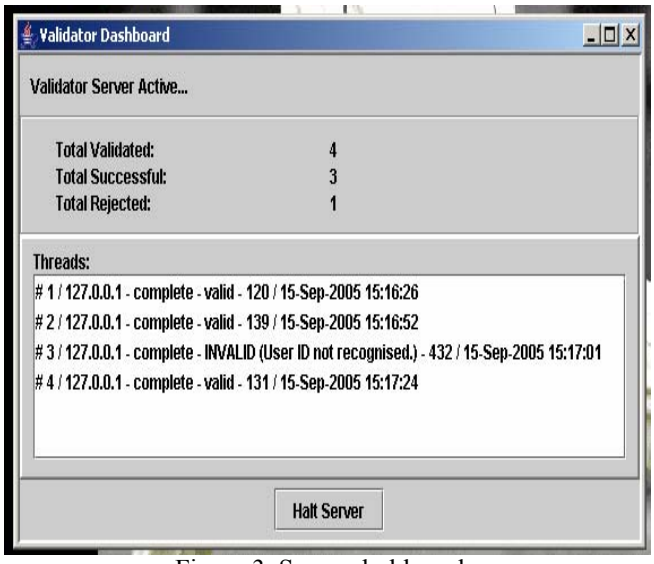

Figure 3. Server dashboard

5.4 Auditing

Auditing offers an effective means of monitoring server behaviour and investigating complaints. Some consideration was given to the issue of trust by requiring the servers with the opportunity to collude, the Validator and the Counter, to record the details and the result of every request processed. Once the election is complete, an auditing team can check the resulting log files. This may involve the investigation of any reported complaints or discrepancies in the count, or the performance of statistical analysis. An extract from the Validator's log file is provided in Figure 4.

```
#1/127.0.0.1-complete-valid-120/15-Sep-2006
14:22:51
#2/127.0.0.1-complete-valid-139/15-Sep-2006
14:23:23
#3/127.0.0.1-complete-INVALID (User ID not
recognised)-sdfg/15-Sep-2005 14:23:39
#4/127.0.0.1-complete-valid-139/15-Sep-2006
14:23:56
```

Figure 4. Log file entry

5.5 System configuration

A separate Java application is provided for the setup of the election. Designed for an administrator, the program can be used to manipulate data relating to the registered list of voters and the list of candidates. The lists will be accessed by the election servers during the voting period.

6. Contextual issues

At a pragmatic level this work has provided an opportunity for exploring many issues. The implemented system satisfies many of the criteria of e-voting and has addressed issues related to privacy, fairness, verifiability, convenience and accountability. Secondary matters such as performance and efficiency have also been taken into account in the design. This work confirms however, the view that it is unpractical for one system to satisfy simultaneously all the e-voting requirements. It is evident that there is a trade-off between authentication and verifiability on one hand and anonymity on the other. Similarly, security and privacy may also appear as conflicting requirements. Although conformance to e-voting criteria was sought through a pragmatic approach, there are some issues, which are partially addressed such as integrity, robustness and other matters which require further investigation.

6.1 Registration and authentication

The issue of registration and authentication are not satisfactorily addressed by the proposed protocol. Indeed most protocols simply assume the existence of an initial voter authentication code or signature, which the voter presents for validation. The FOO protocol for instance, recommends that each voter be issued with a unique private key with which he can form a verifiable signature. That signature can then be attached to the ballot, which is submitted to the Validator. The FOO protocol (amongst others) does not specify how such code or signature is awarded to eligible voters in the first place. Without a secure scheme for distributing signatures, a successful verification of a signature cannot guarantee that the user is eligible. One possible solution is to introduce a preliminary procedure for authentication. The proposed scheme assumes that that the procedure is carried out well in advance of the election and *in person*. All voters would be required to prove their identity to a public official before being given the means to vote online. This issue is at the core of e-voting and may cast doubt on its widespread adoption of e-voting if not dealt with adequately [3, 19].

6.2 Receipt and verifiability

The provision of receipts in the proposed scheme was put forward as a device for challenging incorrect vote recordings. Although voting receipts can be used to verify the accuracy of a vote they leave the voter open to pressure, which may thwart the democratic process. Receipt-freeness has emerged as a unifying concept for expressing the immunity of voters to potential bribery and coercion [20]. The concept was first formulated and implemented in terms of homomorphic schemes [9] and robust versions were also introduced in blind signatures schemes [13]. These schemes tend to be computationally expensive and postulate stronger constraints on the communication channels such as untappability [21]. This might limit their applicability to large-scale elections.

6.3 Anonymous channel

The reliance on an additional server to implement anonymous channels in the prototype may not be very secure. A better solution might be provided by an anonymous mailing system, such as those commonly used in Internet applications. Such a scheme might remove the need for an additional process and thus enhance the robustness of the system. It is generally acknowledged that anonymous channels are difficult to implement. Proposals for alternative implementations include the use of World Wide Web forwarding servers [12], broadcast channels or mixnets [22]. Anonymity can also be enforced through access to Internet cafes or other public services. Anonymous channels are considered a minimal requirement for coercion-resistant schemes [21].

6.4 Robustness

A fundamental weakness of the proposed prototype is its failure to deal with a interruption of the process between the two distinct stages of validation and submission. If for example, a voter loses a signed ballot before submitting it to the Collector, he cannot request a second copy because the Validator assumes that the vote has been sent. The Validator cannot check with the Collector the veracity of the claim; if the ballot has been submitted it will have been anonymised. Nor can the Validator trust the voter since the issue of a new ballot might enable the voter to vote twice. One possible solution would require the Validator to store all of the signed ballots. A voter who loses a ballot could then request the spare copy from the Validator. The Validator can issue the duplicate without allowing the voter to double vote. If the same vote has already been submitted, the Counter will simply discard the duplicate vote. The concentration of many tasks in one server may however increase the vulnerability of the system. This aspect of robustness may be seen as a special case of the situation where the system fails during the voting process. This can be addressed by supporting resume ability explicitly. In the REVS system flexible replication of election servers prevents the occurrence of single points of failure [23].

6.5 System integrity

This research has focussed primarily on the security of the election servers and the channels between client machines and the servers. It has not addressed the issue of security on the client machine from which a vote is cast. A client machine can be subjected to many attacks. Launching a virus to coincide with an election could render thousands

of voter machines unusable, thereby disenfranchising many voters [24]. Malicious software installed without the user's knowledge might reveal the content of a vote or even interfere with the vote before it is submitted. One suggested solution is to personalise the vote list and to design interfaces that operate on codes rather than the choice itself [25]. At system level the reliability of the e-voting scheme depends ultimately on the underlying Internet infrastructure. The partitioning of the underlying network can also prevent a large section of the population from casting their vote.

7. Conclusion

This paper has highlighted the complexity of the deployment of e-voting systems and the inherent security issues that arise from the underlying distributed system. The proposed system has successfully addressed many issues in e-voting and also has pointed out the problems associated with integrity, registration and authentication in particular. The need to reconcile identification and anonymity, on one hand, and verifiability and anonymity on the other hand may be the decisive factor in the wider adoption of e-voting. The challenge is to maintain the integrity of the democratic process by securing eligibility, and preventing bribery and coercion. Moreover, the inability of election authorities to ensure the security and reliability of remote machines may cast doubt on the feasibility of electronic voting as a whole, and may favour the deployment of hybrid systems.

8. References

[1] Voting, Parliamentary Office of Science and Technology, May 2001, *http://www.parliament.uk/post/pn155.pdf.*

[2] Delaune S., Kremer S and Ryan M., Verifying Properties of Electronic-Voting Protocols, *ftp://ftp.cs.bham.ac.uk/pub/authors/M.D.Ryan/06-wote.pdf*

[3] Storer T. and Duncan I., Electronic voting in the UK: Current trends in deployment, requirements and technologies, in *Proceedings of the Third Annual Conference on Privacy, Security and Trust*, 2005.

[4] Kohno T., Stabblefield A., Rubin A. and Wallach D., Analysis of an Electronic System*, IEEE Symposium on Security and Privacy*, 2004.

[5] Penha-Lopes J. M., Why Use an Open Source E-Voting System?, *Tenth Annual Innovation and Technology in Computer Science Education (ITiCSE'05)*, Monte de Caparica, Portugal, June 2005.

[6] Fujioko A., Okamoto T. and Ohta T., A practical Secret Voting Voting Scheme for Large-Scale Elections, *Advances in Cryptology, AUSCRYPT'92,* Springer-Verlag, 1992, pp244-260.

[7] Schneier B., *Applied Cryptography*, John Wiley, 1996.

[8] Tsekmezoglou E.and Illiadis J., A critical View of Voting Technology, *The Electronic Journal for E-Commerce Tools & Applications, Volume 1, Issue 4,* December 2005.

[9] Benaloh J. and Fischer M., A robust and Reliable Cryptographically Secure Election Scheme, *Proceedings of the 16th IEEE Symposium on the Foundations of Computer Science*, Los Angeles, USA, 1985, pp372-382.

[10] Chaum D., Untraceable Electronic Mail, Return Address and Digital Pseudonyms, *Communications of the ACM, 24(2*), 1981, pp103-118.

[11] Chaum D., "Blind Signatures", *Crypto 82*, 1983.

[12] Cranor L.F. and Cyton R.K., A Security-conscious Electronic Polling System for the Internet, *Proceedings of the Hawaii International Conference on System Sciences, 1997, Volume 3*, pp561-570.

[13] Okamoto T., 1997, Receipt-free Electronic Voting Schemes for Large –Scale Elections, *Proceedings of Workshop on Security Protocols '97, LNCS, Vol. 1361,* Springer-Verlag, 1997, pp25-35.

[14] Cramer R., Franklin M., Schoenmakers B. and Yung M., Multi-authority secret ballot elections with linear work, *Advances in Cryptography, EUROCRYPT'96, LNCS, Vol. 1070,* Springer-Verlag, 1996, pp72-83.

[15] Rivest R., Electronic Voting, *ttp://theory.lcs.mit.edu/~rivest/Rivest-ElectronicVoting.pdf*.

[16] Evans D. and Paul N., Election Security: Perception and Reality, *IEEE Security & Privacy, January/February 2004*, pp2-9.

[17] Kofler R. and R. Krimmer and Prosser A., Electronic Voting: Algorithmic and Implementation *Issues 36th Hawaii International Conference on System Sciences (HICSS-36 2003), CD-ROM / Abstracts Proceedings,* Big Island, HI, USA. IEEE Computer Society, 2003.

[18] Kasue S. and Kilian J., Receipt-free Mix-Type Voting Scheme: A Practical Solution to the Implementation of a Voting Booth, *EUROCRYPT'95, LNCS, Vol. 95,* Springer-Verlag, 1995, pp393-403.

[19] Numi H., Salomaa A. and Santean L., Secret Ballot Elections in *Computer Networks, Computers and Security 36(10)*, 1991, pp553-560.

[20] Ku W. and Ho C., An e-voting Scheme Against Bribe and Coercion, *Proceedings of the 2004 IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE;04),* March 2004, pp113-116

[21] Juels A., Catalano and Jakobsson M., Coercion-resistant Electronic Elections, *Workshop on Privacy in the Electronic Society (WPES'05),* Alexandria, Virginia, USA, November 2005, pp61-70.

[22] Furukawa J. and Sako K., An efficient scheme for proving a shuffle. In Kilian J., *CRYPTO '01, LNCS, Vol. 2139,* Springer-Verlag, 2001, pp368-387.

[23] Lebre R., Zuqute A., Joaquim R. and Ferreira P., Internet Voting: Improving Resistance to Malicious Servers in REVS, *IADIS International Conference on Applied Computing*, 2004, Lisbon, March 2004.

[24] Wu C. and Sankaranarayana R., Internet Voting: Concerns and Solutions, Cyber World, 2002, *Proceedings of the First International Symposium First International Symposium on Cyber Worlds (CW'02),* 2002, pp261- 266.

[25] Breunesse C., Jacobs B. and Oostdjk M., Voting using Java Card smart cards: A case study, *http://citeseer.ist.psu.edu/breunesse02voting.html*, 2002.