Robust e-Voting Composition

Richard Cooke\textsuperscript{1} and Rachid Anane\textsuperscript{2}

\textsuperscript{1}School of Computer Science, University of Birmingham\newline r.cooke@cs.bham.ac.uk
\textsuperscript{2}Faculty of Engineering and Computing, Coventry University\newline r.anane@coventry.ac.uk

ABSTRACT
This paper is concerned with the presentation of a perspective on robustness in e-voting systems. It is argued that the effective design of an e-voting system and its viability can be enhanced by a two-pronged approach to robustness. First, it requires a clear distinction between two forms of robustness: robustness at protocol level and robustness at system level. Second, selected technologies should be integrated into an appropriate architecture in order to address robustness simultaneously at the two levels. The proposed approach is illustrated by the design and implementation of an e-voting system based on the FOO92 protocol. A service-oriented architecture supported by onion routing forms the basis of the system. It facilitates the distribution of tasks and state, the dynamic path configuration and the just-in-time (JIT) composition of the election system. The system conforms to most e-voting requirements.

Keywords: Web service composition, JIT approach, blind signatures, FOO protocol, onion routing.

1 Introduction
The present consensus on the identification of desirable e-voting properties has settled on four criteria for evaluation: integrity, privacy, verifiability and robustness. In the fulfilment of these requirements two distinct levels of concern are identified, a protocol level and a system level. The protocol level is concerned with the deployment and the behaviour of the e-voting protocol as an application for conducting elections; the system level relates to the underlying network, the implementation of the servers and their interaction. This dichotomy is a reminder that a remote voting system is essentially an application supported by a distributed system.

Integrity and verifiability are issues that relate primarily to the protocol level. Privacy, on the other hand, is meaningful at protocol level but its full satisfaction, especially anonymity, requires an awareness of the characteristics of the underlying distributed system. Although secrecy can be achieved by cryptographic and therefore mathematical methods only, anonymity requires the marshalling of distributed technologies such as mixnets [1, 2], onion routing [3] or Web servers [4]. Anonymity is an issue that straddles both levels, a characteristic that it shares with the management of denial of service (DOS).

The viability of an e-voting system requires, at protocol level, conformance to at least the integrity and the privacy requirements. It also assumes, at system level, availability and resilience when subjected to malicious attacks or when faults occur. It is evident that the soundness of the underlying distributed system determines the utility of an e-voting system. In this respect, robustness may be defined generically as the ability of a system to deal effectively with unexpected input or behaviour, large volumes of data, and to continue providing services in conformance with stated requirements. In most of the published research on e-voting systems, robustness is often confined to the protocol level while issues that pertain to the distributed systems level are seldom explored.

This paper is concerned with the presentation of an approach that promotes a comprehensive view of robustness and a selective application of distributed systems technologies. It is argued that this two-pronged approach can successfully address robustness issues that straddle the two levels such as anonymity and denial of service, through the selection of technologies and their integration into an appropriate architecture. The approach is illustrated by the development of a realistic e-voting system based on the implementation of the FOO92 protocol [5]. A service-oriented architecture was adopted which facilitates the JIT dynamic composition of Web services, decentralisation, distribution of state and tasks, and onion routing.

The remainder of the paper is organised as follows. Section 2 identifies the main requirements of e-voting systems. Section 3 presents a perspective on robustness. Section 4 gives an outline of the e-voting system and Section 5 details its implementation as a service-oriented architecture. Section 6 offers an evaluation of the approach and of the system, and Section 7 concludes the paper.

2 e-Voting
As the deployment of e-voting systems can potentially have a significant impact on the political, economic and social domains, conformance to specific requirements has become a critical test. At the core of voting systems lies the need for compliance with the democratic
process by ensuring the integrity and the privacy of votes through four specific stages performed by specific election authorities:
- The registration of eligible voters (Administrator).
- The validation of potential voters (Validator).
- The collection of the votes (Collector).
- The tallying or counting of the votes (Counter).

The formulation of e-voting requirements has led to a refinement of criteria and has become a research area in its own right [6, 7]. The main e-voting desirable properties can be identified as follows:

1. **Integrity**: this is concerned with the property that the different stages that process votes do not alter them or corrupt them. More specifically, integrity implies that a vote is cast as intended, recorded as cast and counted as recorded.

2. **Privacy**: this criterion is aimed at ensuring that votes are cast anonymously, namely that it is not possible to associate a vote with the corresponding voter (untraceability), and that the vote is secret. Another aspect of privacy concerns the inability of voters to demonstrate that they have voted in a particular way, and their ability to withstand coercive measures (coercion resistance).

3. **Verifiability**: this property refers to the openness of the system to formal and practical scrutiny and is related to integrity. It should be possible for voters to check that their vote was correctly recorded (individual verifiability) and that all the votes were processed and counted correctly (universal verifiability).

4. **Robustness**: this is defined as the resilience of the system when cheating behaviour is detected, partial component malfunction occurs or when it is subjected to external malicious attacks.

In e-voting research the focus has been on the conformance to integrity, privacy and verifiability. Despite its crucial importance robustness is often implicit or it refers to issues that pertain to undifferentiated levels of concern. As most e-voting schemes are deployed as distributed systems, robustness is bound to involve many facets.

In many e-voting systems the perspective on robustness is quite broad and lacks focus. It covers issues that are associated with the e-voting application, such as cheating, and those that may occur in the underlying distributed system such as denial of service. It is evident that there is a semantic difference in behaviour between an election authority that attempts to cheat and the faulty server that implements it.

3 Robustness

A distinction between an application and the system that serves it suggests an appreciation of robustness at two levels: the first level concerns protocol behaviour and the second concerns the underlying distributed system. The generic definition of robustness can therefore be refined to capture the main issues at the two levels:

**Robustness at protocol level**: all agencies contribute to the voting process to satisfy stated requirements; measures are in place to discourage and hinder cheating behaviour and no agency or group of agencies can attempt to thwart the democratic process without being detected.

**Robustness at system level**: a system should deal effectively with unexpected input or volume of data, detect and recover from failures or malicious attacks and continue to provide services in conformance with stated requirements. Robustness at system level is a prerequisite for robustness at protocol level. For example, insecure servers or channels cannot ensure the privacy of the voter or the secrecy of the vote.

3.1 Robustness in e-voting systems

The range of robustness issues in e-voting is best investigated by reviewing the properties of some e-voting systems. Sensus [4] presents the issue of robustness under the ‘democracy’ criterion, but provides limited support for it. Sensus assumes that communication occurs over anonymous channels, and does not address robustness at system level. SEAS [8] is presented as an improvement to Sensus by preventing the Validator from voting on behalf of eligible voters who abstained, an issue relevant to protocol level.

REVS was designed with robustness as a key property of the system, and considers robustness at two levels [9]. REVS deals with robustness at system level by replicating servers and maintaining state information and ensuring availability in case of interruption. At protocol level, the system is evaluated against the integrity criterion.

In Prêt à voter, robustness is considered implicitly as the ability of the system to cope with, for example, the cheating behaviour of the mix servers. This is achieved by the removal of a cheating mix server and its simulation by a quorum Q of other mix servers [10]. There is however no explicit distinction between the different levels of robustness.

The design of Civitas [11] is motivated by the need to achieve full conformance to e-voting requirements, especially coercion resistance. Robustness is addressed explicitly and is expressed in terms of trust, namely the ability of a legitimate voter to cast a vote without coercion and to have the vote cast as intended, recorded as cast and counted as cast. Helios [12] does not address robustness explicitly at any level, and relies on extensive auditing and verification to detect malicious behaviour and ensure conformance to integrity. Implicitly the focus is at protocol level.

This brief survey reveals that most of these systems address robustness mainly at protocol level, either implicitly or explicitly as in Civitas. REVS appears as one of the few exceptions where robustness is considered at two levels without explicit differentiation.
3.2 Robustness in distributed systems

The rationale for considering robustness at two levels stems from the realisation that the ability of an e-voting application to deal with unexpected situations depends on the flexibility of the underlying distributed system and the choice of technology. Client-server architectures may be easy to implement but a server may be a single point of failure; P2P systems offer more flexibility and resilience but critical interactions require a trusted environment; Web services may be simpler and easier to integrate but require verbose and inefficient encoding; stateful servers are more convenient but require the active maintenance of a consistent state. Most significantly, however, networks and servers can be subjected to denial of service (DOS) attacks [13]. Typically these attacks involve either swamping the network with garbage messages or overloading a server with computationally intensive and useless requests.

4 An e-Voting system

The proposed approach is illustrated by the design and implementation of an electronic voting system. It is used as a vehicle for exploring the issue of robustness and in particular the impact of selected technologies and architecture on robustness at the two levels. The proposed approach is based on two premises: 1) a sound implementation of an e-voting system requires the sound implementation of the underlying distributed system and 2) a holistic design and implementation approach that addresses both levels simultaneously offers more resilience and lead to better integration.

In the design of the e-voting system it is assumed that the registration of voters is done before the election, and that voters obtain their initial codes through out-of-band authentication. It is also assumed that the voter’s machine is reliable and trusted and that votes are cast as intended. This is in line with the rationale that underpins the design of the Helios system, namely that coercion resistance may not be a critical issue because of the context and the limited scope of the deployment of the system.

4.1 FOO92 Protocol

The proposed perspective on the impact of engineering solutions on the robustness of an e-voting system takes the implementation of the FOO92 protocol as a foundation for a case study. Thanks to its flexibility, its efficiency and its conformance to most e-voting criteria, the FOO92 protocol has formed the basis of many voting protocols and has been the subject of various enhancements [14]. The protocol is based on blind signatures [15], and models the voting process as follows:

1. Voter retrieves ballot from Administrator
2. Voter completes the ballot and blinds it.
3. Voter constructs a message with the ballot and his identity and encrypts it with the Validator’s public key.
4. Voter sends the message to Validator.
5. Validator decrypts the message, validates Voter and signs the ballot.
6. Validator returns the blinded ballot to Voter.
7. Voter unblinds the ballot and encrypts it with Counter’s public key.
8. Voter forwards the ballot to Counter over an anonymous channel, through Collector.
9. Counter checks for Validator’s signature, decrypts ballot and increments the corresponding counts.

The development of the system involves essentially mapping the election authorities to specific servers, providing support for validation and authentication, setting up anonymous channels and recording votes accurately. The design of the system should also cater for the requirements of robustness.

4.2 The voting process

The system architecture that supports the voting process is shown in Figure 1. The processing of the voter and the corresponding vote information identifies three distinct virtual spaces in the diagram: validation where the voter is known but the vote is not known; transmission of the vote where the voter is not known and the vote is not known, and recording of the vote where the voter is not known and the vote is known.

Validation: (voter is known and the vote is not known) this phase is concerned with the authentication of the voter by the Administrator and the retrieval of election details and identification information (Step 1, 2). A vote with a personal random number (RN) is blinded and sent to the Validator (Step 3). The validation of the voter through its alias, VT1, involves checking its credentials against the electoral roll (e-roll nodes) (Step 4, 5) and determining whether they have already been validated (Step 6, 7). If the voter is eligible the blinded vote is signed and returned to the voter by the Validator (Step 8).

Transmission: (voter is not known and the vote is not known) on successful validation, the voter generates a message which includes the random number, the vote and election details and encrypts it with the public key of the Counter (v = {{choice, electionId, RN}val-priv}count-pub). It then transmits the message to the Counter via a chain of routing nodes (Step 9, 10, 11, 12) and the Collector. The chain acts as an anonymous channel. On receipt of the information, the Collector extracts the packaged ballot, checks its validity and forwards it to the Counter (Step 13).

Recording: (voter is not known and the vote is known) the Counter checks the ballot for validation by the Validator, extracts the vote and adds it to the appropriate tally. The vote is recorded in the database against the personal random number (RN) of the Voter.

4.3 Secure and anonymous processing

In the diagram in Figure 1 a verbose notation for mathematical expressions was used for the sake of clarity. In the exchange of messages, secure and anonymous transmission is achieved by:
Figure 1. e-Voting process
The generation of a random number by the voter as a unique identification token. This number facilitates individual verifiability and prevents multiple votes.

The anonymity and secrecy of the vote is achieved in two ways, by blinding signatures and by asymmetric encryption. The is illustrated by the message sent to the Validator by the Voter $$\text{[}[[\text{choice, electionId, RN}]\text{blinded}, \text{VT1, electionId, voter-pub}]\text{val-pub}]$$.

The asymmetric encryption where messages are encrypted for secrecy using the public key of a server (Counter), or signed by a server (Validator) with its private key $$\text{[}[[\text{choice, electionId, RN}]\text{blinded}]\text{val-priv}\text{voter-pub}]$$.

The onion-routing itinerary is generated randomly and transmitted to the voter with the election details. It is designed to support anonymous communication. In the proposed system a Tor-like circuit [16] is built by the Administrator randomly from a set of available nodes $$(N,[[\text{col}N3\text{-pub}, N3]N2\text{-pub}, N2]N1\text{-pub})$$ and passed to the Voter. Although three nodes are used the circuit length is variable. The innermost node of the circuit is the Collector and is encrypted with the public key of the node preceding it (N3); at the next layer the encrypted innermost node and the node preceding it are encrypted with the public key of the node preceding the preceding node (N2) and so on until the outer node is encrypted. Only routing node N1 on the outer layer is known to the Voter. The Voter constructs a message which includes the vote and the path $$(v, [[\text{collector}N3\text{-pub}, N3]N2\text{-pub}, N2]N1\text{-pub }]N1\text{-pub})$$. When it receives the message N1 decrypts it to access the encrypted route and to determine the next node in the network $$[[\text{collect}N3\text{-pub}, N3]N2\text{-pub}]N2\text{-pub}].$$ This forms an important component of the JIT approach.

Each server contributes to the monitoring of the voting process, by logging and signing explicitly its transactions and by signing and authenticating messages where appropriate. Unauthenticated messages are discarded in order to minimise overload on the network and on the Counter.

5.1 Architecture

The service-oriented architecture is composed of a number of key services:

Administrator Service: provides a user interface to specify the election; coordinates the agencies used in an election; serves the Voter Client to the voter and publishes the results when an election ends.

Validator Service: receives the blinded ballot from the Voter Client, using the alias provided by the voter; checks if the voter exists and if it has not been already validated.

Collector Service: receives the validated ballot from the Voter Client; signs and forwards the ballot to the Counter.

Counter Service: receives the ballot from the Collector; checks its signature; extracts and records the personal random number and adds the vote to the tally.

Routing Nodes: receives a ballot either directly from a Voter Client or via a routing node; decrypts the routing path and determines the following agency in the path.

Voter Client: an applet used for casting a vote.

The Administrator is the most important service in the system. It is a trusted election authority that initiates and coordinates elections. Conceptually it includes seven key components: the administrator user interface (UI), the public user interface, the public Web service interface, the check voter Web service interface, the election coordination, the agency monitoring and the persistence layer.

5.2 Composition process

At the start of the election the Administrator selects a Validator, a Collector and a Counter at random from the
agencies that are online and are not used by another election (Figure 2). Once selected, these agencies are notified and given the election id and the details of the relevant nodes in the system. For example, once the Administrator has built the network of e-roll nodes, it will inform the Validator about the location of the electoral roll nodes and their public keys.

The Collector and Counter are given the details of the public key of the Validator so they can check that the ballot validation signature is correct. The process for composing the electoral roll agencies is similar, except that the Administrator will attempt to compose the electoral roll agencies as requested by the election official. If there are not enough available agencies, the administrator will simply compose a reduced number of nodes. The composition process is another manifestation of the JIT approach.

5.3 Cryptography

The Secure Socket Layer (SSL) was deemed unsuitable for secure communication as it only provides point to point security. Specific cryptography functions had to be implemented. These include methods for key generation, key storage, encrypting XML elements, and decrypting XML elements, signing XML elements and verifying the signatures added to XML elements. A hybrid cryptosystem was used for sending efficiently and securely messages between the agencies. Each message is encrypted using a freshly generated symmetric AES-128 key. This plain key is then encrypted using the RSA-2048 public key of the recipient and forwarded along with the message. When it receives the message the recipient decrypts the encrypted symmetric key with its private key to retrieve the symmetric key and decrypts the content of the message. The public keys are distributed to agencies when they are setup as X.509 certificates stored in the key store. An encrypted XML message is shown below:

```xml
<encrypted-message>
  <sym-key>esf234tr4tg4t23gov6</sym-key>
  <encrypted-content>f4wrt3rt5egbdvbvt5Y2r3vevsf435gd</encrypted-content>
</encrypted-message>
```

5.4 Implementation issues

Java was chosen as the programming language for the implementation of the system. It is suitable for Web development and benefits from the availability of libraries for Web services and cryptography. All the application logic was written in EJBs and Glass Fish 3.0 was adopted as the application container. EJBs provide many transparent services such as transactions, security, and pooling and thread safety. The Web services were built with the JAXWS framework. A simple web service is outlined below:

```java
@WebService()
public class VoterClient {
  @EJB
  private BallotGenerationManager ballotGenerator;
  /**
   * Web service operation *
   */
  @WebMethod(operationName = "getBallot")
  public String getBallot(@WebParam(name="getBallot")
                            String getBallot) {
    return ballotGenerator.getBallot(voterId, voterCode);
  }
}
```

With the deployment of the Web service to the application server the JAX-WS framework generates a Web service stub for this service and publishes a WSDL file.

Data management was supported by the design and implementation of a MySQL relational database. The Java Persistence API was used to implement Object Relation mapping between Java objects and the relational database tables.

6 Evaluation

The evaluation is concerned with the extent to which the system conforms to e-voting requirements in general and robustness in particular.

6.1 Verifiability

Individual verifiability and universal verifiability are supported by the accurate recording of the votes and their subsequent publication without compromising privacy. Whilst individual verifiability can be performed immediately after a vote is cast, universal verifiability, for the sake of fairness, is enabled only at the end of the election. Verifiability can be conducted by the stakeholders and the voter in particular, by using their personal random number (Figure 3). Through verifiability and auditing voters can contribute significantly to the robustness of the system at protocol level. This approach is strongly promoted in Helios.

![Individual verifiability](image)

**Figure 3. Individual verifiability**
6.2 Integrity
The Administrator is performing a number of critical functions under the fundamental assumption that it is trusted. Other agencies however need to be monitored and their behaviour constrained. Some potential cases of misbehaviour are considered below.

It would be difficult for a Validator to vote on behalf of a voter who abstained. The use of aliases [17] and the distribution of the electoral roll across many nodes are designed to prevent the Validator from identifying the voters who abstain. Although it can always create a new identifier, the alias will not be cleared by the Administrator and will therefore lead to discrepancies in the tallies of the votes. As for the Collector, without collusion, it cannot forge or modify votes since they must be signed by the Validator. It can however drop votes but this can be detected through verifiability and tallying. A Counter may be able to add spurious votes but this can be detected since the Administrator is keeping track of the total number of validated voters, which should be greater than or equal to the tally of the votes produced by the Counter. Modification of votes by the Counter is hindered by verification by voters. The recording of the random number in the Counter allows for votes to be computed accurately and to prevent multiple votes by voters.

Besides potential individual misbehaviour, collusion between agencies is another cause for concern. The transient configurations that the JIT approach generates can be an obstacle to the collusion between servers. The use of onion routing ensures that votes arrive to the Collector from different routes. Votes are only accepted by the Counter if they are sent and signed by the Collector. The injection of spurious votes from entities outside the e-voting system is made difficult by the partial obfuscation of the network.

The integrity of the system is also enhanced by the logs of server transactions and the monitoring of server activity. The combination of server monitoring and voter verification can help detect malicious behaviour by servers and voters. It is difficult for a server to drop, add or modify votes without being detected. Thanks to the Web Services framework it is possible to replace a server if it is faulty or is dishonest. All these measures contribute to robustness at protocol level.

6.3 Privacy
The system provides support for the anonymity of the voters and the secrecy of the vote through a combination of asymmetric encryption and blinding scheme. Privacy requirements and anonymity in particular, are also addressed by architectural features such as onion routing and dynamic routing allocation. The onion routing approach was considered more suitable than mixnets [2], thanks to its greater impact. A combination of the two schemes is however a feasible option. Privacy is also ensured by implementing one-way communications especially in the last two virtual spaces of the voting process. It is further supported by the generation of a random number by the voter and its inclusion with the ballot rather than the reliance on the production and transmission of a receipt by the Counter.

Coercion resistance
There is a potential conflict between coercion-resistance and individual verifiability in e-voting schemes. The ability of voters to check that their vote was recorded accurately may make them vulnerable to coercion. It has been argued that in some voting contexts coercion resistance may not be a fundamental requirement [12]. This is particularly applicable to student elections and online specialist communities such as ACM and IEEE. Helios is a system where the viability of a system does not depend on coercion-resistance. This characteristic is common to many implementations of the FOO92 protocol such as Sensus [4], SEAS [9] and REVS [10].

6.4 Robustness
Although e-voting systems are designed to serve voters, their intervention may be a source of insecurity. Voters are not trusted because of their autonomy and the arbitrary and unsupervised nature of their intervention. The impact of malicious attacks is however mitigated by the dynamic generation of the routing nodes and the absence of obvious patterns in the composition of the system. Only legitimate voters are given a path to the servers and access is granted to the first routing node only.

Denial of service
Denial-of-service attacks can take different forms and many countermeasures were proposed to deal with these attacks at different levels. Some systems rely on filters on the network for detecting and blocking DOS attacks [18]. At the operating system (OS) level protocols can be configured to deter DOS attacks. DOS attacks rely on the knowledge of the architecture of the networks and of the servers. The resilience of the proposed system is underpinned by this assumption. For its defence against DOS attacks the proposed system relies primarily on the just-in-time (JIT) composition of the election system, its flexible configuration with different onion-routed nodes, as well as the limited knowledge of the network structure and of the servers by the different election authorities and by the voter. This can be an impediment to mapping out the network structure and mount concerted DOS attacks. The task is also supported by the active monitoring of the servers. The ability to replace defective or rogue servers by new ones can also limit the impact of the DOS attacks.

Reliability
The JIT approach enhances reliability: only working and available servers are selected. Furthermore at a lower level, the chain of forwarding servers can be constructed adaptively in such a way as to minimise network congestion and to overcome network partitions.

In the proposed system, no state is maintained on the intermediate servers, namely the Validator, routing
nodes and the Collector. The distributed electoral rolls (e-rolls), the Administrator and the Counter maintain the state that underpins the functionality and viability of the system. Strong backup is required in the case of these stateful servers. However, the partial statelessness of the system and the flexibility of Web services allow for mediating servers to be replaced easily. In REVOS on the other hand, the resilience of the system is ensured by replicating servers and by maintaining state on the participation of the voters.

6.5 Contribution
The main contribution of this paper lies in the distinction between two types of robustness, and in the design of a service-oriented architecture and the selection of appropriate technologies to address robustness at two levels. For example, in addition to support for anonymity and dynamic configuration, onion routing can act as an obstacle to DOS attacks.

The design of the distributed system was guided by distributed trust concepts [6], separation of concerns and processing to prevent collusion (different servers), partial access to state information to prevent cheating and processing to prevent collusion (different servers), the use of aliases as well as the active monitoring of server activity, its aim is to promote the design of resilient e-voting systems. The approach is supported by the implementation of the servers as Web services and their composition. This perspective on robustness offers scope for wider decentralisation and scalability, and invites a more integrated and holistic approach to the development of e-voting systems. It contributes ultimately to the satisfaction of most e-voting requirements.

7 Conclusion
In this paper the implementation of the FOO92 protocol was used as a vehicle for presenting a perspective on robustness in e-voting systems. It advocates a distinction between robustness at protocol level and robustness at system level, and the identification of suitable architecture and technologies for addressing the two levels simultaneously. With its emphasis on the distribution of state, the decentralisation of tasks, the JIT routing configuration, the use of aliases as well as the active monitoring of server activity, its aim is to promote the design of resilient e-voting systems. The approach is supported by the implementation of the servers as Web services and their composition. This perspective on robustness offers scope for wider decentralisation and scalability, and invites a more integrated and holistic approach to the development of e-voting systems. It contributes ultimately to the satisfaction of most e-voting requirements.

8 References