A Service-Oriented Architecture for Robust e-Voting

Richard Cooke[1] and Rachid Anane[2]

[1]*School of Computer Science, University of Birmingham*
r.cooke@cs.bham.ac.uk
[2]*Faculty of Engineering and Computing, Coventry University*
r.anane@coventry.ac.uk

ABSTRACT

Of all the requirements for e-voting systems robustness is the one that has received the least attention. This paper is concerned with addressing this issue. It is argued that a two-level consideration of robustness can facilitate the design of e-voting systems and enhance their resilience. An approach is proposed which requires, as a first step, an explicit awareness of robustness at protocol level and robustness at system level. The second step involves the identification of appropriate technologies and their integration into an architecture where the two forms of robustness are addressed. The approach is illustrated by the design and implementation of a service-oriented architecture for robust e-voting (SOREV), based on the FOO92 protocol. The service-oriented architecture has provided the framework for the integration of selected technologies such as blind signatures, encryption and onion routing. In addition to the Just-in-time (JIT) composition of the e-voting system, it supports the distribution of tasks and state. The system conforms to most e-voting requirements.

Keywords: Robustness, Web services composition, JIT approach, blind signatures, FOO92 protocol, onion routing.

1 Introduction

The viability of e-voting systems is often assessed by their conformance to agreed and stated requirements. This fundamental constraint is driving the investigation into requirements and verification [1]**.** The present consensus on the identification of desirable e-voting properties has settled on four main criteria: 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 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 remote voting systems are essentially applications supported by distributed systems.

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, may require 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 mix nets [2, 3], onion routing [4] or Web servers [5]. Anonymity is an issue that straddles different levels, a characteristic that it shares with the management of denial of service (DOS).

An e-voting system is expected to conform, at protocol level to at least the integrity and the privacy requirements. It also assumes at distributed system level 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. This therefore qualifies robustness as a property that spans both levels of concerns. 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 a service in conformance with stated requirements. In published research on e-voting systems, robustness is often confined to the protocol level, whilst issues that pertain to distributed system level are seldom explored.

This paper is concerned with the presentation of an approach that promotes a more focused view of robustness in e-voting systems, and a selective application of distributed systems technologies in the development of robust systems. It is argued that this two-pronged strategy can successfully address robustness issues at protocol level and at system level.

The contribution of this paper lies in the explicit differentiation between two forms of robustness and the integration of various technologies into an appropriate architecture to address this issue. The approach is illustrated by the design and implementation of a service-oriented architecture for robust e-voting (SOREV), based on the FOO92 protocol [6]. It is supported by the dynamic composition of Web services, the distribution of state and tasks, and by 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 particular perspective on robustness in e-voting systems. Section 4 gives an outline of the behaviour of the proposed system and Section 5 details the implementation of the corresponding service-oriented architecture. Section 6 offers an evaluation of the approach and of the system, in context with other e-voting systems. Section 7 puts

forward some pointers for further work, and Section 8 concludes the paper.

2 e-Voting

With the increasing interest in the deployment of e-voting systems and the potentially significant impact they can have 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 its viability through four 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 voting process is conducted within specific constraints. A secure electronic voting scheme must meet the following theoretical requirements [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 votes have been counted.

2.1 Advantages of e-voting

The exponents of e-voting often put forward a number of reasons for promoting its wider deployment. They contend that it has a number of benefits:

Participation: electronic voting has the potential of appealing to a wider section of the population. An Internet-based system will enhance convenience and flexibility. Voters will be able to cast their vote anytime and anywhere.

Efficiency and accuracy: e-voting promises to improve the accuracy of the voting process at various stages. Computerisation and network technology, it is argued, will improve efficiency in processing votes and will lead to quicker results.

Transparency: an e-voting system will lead to greater openness to public scrutiny and greater accountability. The scrutiny should apply to the source code by experts as well as verifiability of votes by voters. This demand for openness should not be achieved at the expense of security.

2.2 e-Voting properties

The formulation of e-voting requirements has led to a refinement of criteria and has become an important research area in its own right [8, 9]. The most important e-voting properties can be grouped as follows:

- **Integrity:** this is concerned with the property that the different agencies that process votes do not alter them or corrupt them, and intruders do not interfere with the voting process. It also refers to the ability of eligible voters to vote and to vote only once. More specifically, integrity implies that a vote is cast as intended, recorded as cast and counted as recorded. Integrity entails honest behaviour and collusion resistance.
- **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).
- **Verifiability:** this 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 votes were correctly recorded (individual verifiability) and that all the votes were processed and counted correctly (universal verifiability). It is believed that with enhanced verifiability voters will have more confidence in the conduct of remote elections and in their results.
- **Robustness:** this is defined as the resilience of the system when cheating behaviour is detected, partial component or system malfunction occurs or when it is subjected to external malicious attacks. The system should operate as expected in abnormal conditions or in a hostile environment.

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

In the Prêt à voter system, for example, robustness is concerned 'with the resilience in the face of random faults, as well as deliberate attempts to disrupt the election, such as denial of service' [10]. It is seen as an indication of the ability of the implementation of the protocol to deal with unexpected input, faults or cheating by an election authority.

This perspective on robustness is quite broad and lacks focus. It covers issues that pertain specifically to the e-voting application, such as cheating, and those that may occur in the underlying distributed system such as faults and 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.

2.3 e-voting protocols

In most e-voting systems the design of protocols is driven by two major concerns, which identify two main virtual spaces:

1. Ensuring that if the voter is known the vote is not known
2. Ensuring that if the vote is known the voter is not known.

It is this complete dissociation between a vote and the voter who cast it, namely anonymity or untraceability, which lends credibility to an e-voting system. E-voting systems can be classified according to the way they implement it. Three main schemes were devised to support it:

- Schemes based on homomorphic encryption reduce a ballot to a number and ensure that all the voter choices are kept secret [11]. This has the advantage that the vote can be performed without decrypting any of the ballots. It is however computationally expensive.
- Schemes that generate mixes (mix nets) permute different entities to hide the correspondence between input and output items, and ensure that an item is only processed once [2]. The connection between voters and ballots is difficult to establish.
- Schemes based on blind signatures [6, 12] allow an agency to sign a message without knowing its contents. Schemes based on blind signatures present a number of advantages. 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 Robustness

A distinction between an application and the system that serves it is useful for a clear identification of the issues related to robustness. This consideration suggests an appreciation of robustness at two different levels. It promotes an awareness of issues at protocol level such as integrity and privacy and those that pertain to the distributed system, such as resilience and scalability. It is also helpful in identifying more refined requirements and promoting appropriate configurations. The generic definition of robustness can be refined to capture the distinctive features of robustness at the two levels:

Robustness at protocol level is defined as the capability of the application to ensure that the privacy, integrity and verifiability of the e-voting process are preserved, when faced with incorrect procedures or malicious behaviour and attacks. Robustness at this level is seen as an overarching concept that helps evaluate the ability of a system to conform to e-voting requirements. 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.

An additional component of robustness at protocol level is **recoverability**. It is defined as the capability of the e-voting application to recover from malfunctions by restoring the state of the voting process, and by resuming successfully its operations [13].

Robustness at system level is defined as the capability of the distributed system to support the functions of the e-voting application when faced with component failure or abnormal behaviour. The system should deal effectively with unexpected input or volume of data, and detect and recover from malicious attacks. It should continue to provide a service in conformance with stated requirements. Robustness at this 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.
Although robustness of e-voting systems at system level can be expressed in terms of many properties [14], resilience, scalability, flexibility and cost were deemed particularly relevant:
Resilience: the capability of a system to mitigate the impact of abnormal conditions and failures on its behaviour.
Scalability: the capability of a system to accommodate growth and to deal effectively and adaptively with fluctuations in load.
Flexibility: capability of a system to operate in different platforms and environments and to support interoperability and various configurations.
Cost: an evaluation of the resources used in the system, the mode of interaction of the components and the implied processing involved in its operation. Implementing a robust system may be expensive because of the replication of resources and heavy communications required.

3.1 Robustness in e-voting systems

The range of robustness issues that arise in e-voting can be partially uncovered by reviewing the properties of some e-voting systems. Sensus [5] presents the issue of robustness under the ‘democracy’ criterion, but provides limited support for it. In Sensus it is assumed that communication occurs over anonymous channels. The issue of robustness is not considered explicitly, and robustness at system level is not addressed. SEAS [15] is presented as an improvement to Sensus in preventing the Validator from voting on behalf of eligible voters who abstain. It implicitly enhances the robustness of the e-voting system at protocol level.

REVS was designed with robustness as a key property of the system, and considers robustness at two levels [16]. REVS deals with robustness at system level by replicating servers, maintaining state information and ensuring ‘resumability’ in case of interruption. At protocol level, the system is evaluated against the

integrity criterion. The provision of many servers which contribute to a quorum policy is considered an impediment to collusion. REVS makes use of redundant information and servers to improve robustness.

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]. Other aspects of robustness are considered as part of the practical implementation at system level. There is however no explicit distinction between the different levels of robustness.

The design of Civitas [17] 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. The focus is on robustness at protocol level; issues that relate to the underlying distributed system are identified as limitations for further work. Helios [18] 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 the protocol level.

This brief review 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 exceptions where robustness is considered at two levels without an explicit differentiation. It can be argued however that the design of some of these systems may be motivated by transparency considerations.

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 underlining distributed system and on the choice of technology**.** The selection of a client-server or P2P architecture, the inclusion of stateful or stateless servers, the reliance on Web services or distributed object middleware are all implementation decisions that affect the robustness of the distributed systems, and by implication that of the application itself. Client-server architectures may be easy to implement but the server may be a single point of failure; P2P systems offer more flexibility and resilience but critical interactions require a trustworthy environment; Web services may be simpler and easier to integrate but require verbose and inefficient encoding; stateful servers offer more convenience but require the active maintenance of a consistent state. Most significantly, however, networks and servers can be subjected to denial of service (DOS) attacks [19]. Typically, these attacks involve either swamping the network with garbage messages or overloading a server with computationally intensive and useless requests. In both cases the aim is to prevent the system from performing its role effectively.

Various methods were proposed for enhancing the robustness of a distributed system: distribution of tasks and state, replication of resources and servers, provision of flexible routes and inclusion of stateless protocols and servers.

4 An e-Voting system

The proposed approach is illustrated by the design and implementation of an electronic voting system. It will be 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 application 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.

The scope of the investigation of robustness will be confined to specific issues. At system level, the main concern will be denial of service and faulty servers. At protocol level, the focus will be on integrity issues such as cheating and collusion, and privacy issues such as anonymity and secrecy. 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 codes and aliases through out-of-band authentication. It is also assumed that the voter's machine is reliable and trustworthy and that votes are cast as intended. In line with the rationale that underpins the design of the Helios system, coercion resistance is not considered a critical issue because of the context and the limited scope of the deployment of the system.

4.1 FOO92 Protocol

An implementation of the FOO92 protocol is used as a basis for a case study of the impact of engineering solutions on the robustness of an e-voting system. Thanks to its flexibility, its efficiency and its conformance to most e-voting criteria, the FOO92 protocol has formed the basis for many voting protocols and has been the subject of various enhancements. It has also a high degree of compatibility with manual systems [20]. The FOO92 protocol has the advantage of simplicity, offers a clear separation between concerns and can accommodate flexible implementations at distributed system level. The protocol is based on blind signatures [12] 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 containing 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 on the ballot, decrypts it and increments the corresponding count.
The development of the system involves essentially mapping the election authorities to specific servers, providing support for validation and authentication through access to the electoral roll, setting up anonymous channels and recording votes. The design of the system should also cater for the requirements of robustness at two levels.

4.2 The voting process

The system architecture that supports the voting process is shown in **Fig.** 1. It incorporates all the servers and their modes of interaction.

The processing of the 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: the voter is known and the vote is not known. This phase is concerned with 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) and election details is blinded and sent to the Validator (Step 3). With blind signatures the Validator signs the message sent by the Voter without being able to read its content [17]. The validation of the voter through its alias VT1 involves checking its credentials against the electoral roll (e-roll nodes) (Steps 4, 5) and determining whether they have already been validated (Steps 6, 7). If the voter is eligible the blinded vote is signed and returned to the voter by the Validator (Step 8). The Validator requires an acknowledgment from the Voter in order to prevent multiple requests for validation from the same voter.

Transmission: the voter is not known and the vote is not known. On successful validation, the voter unblinds the message signed by the Validator and encrypts it with the public key of the Counter ($V=\{\{choice, electionId, RN\}_{val\text{-}priv}\}_{count\text{-}pub}$). It then transmits the message to the Counter via a chain of routing nodes (Steps 9, 10, 11, 12) and the Collector ($V, \{\{col\}_{N3\text{-}pub}, N3\}_{N2\text{-}pub}, N2\}_{N1\text{-}pub}$). The chain acts as an anonymous channel. On receipt of the message, the Collector extracts the packaged ballot, checks its validity and forwards it to the Counter (Step 13).

Recording: the 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 also recorded in the database against the personal random number of the Voter.

4.3 Secure and anonymous processing

The notation used in **Fig.** 1 includes the application of asymmetric encryption to the messages. In the exchange of these messages, secure and anonymous transmission is achieved by:

- The generation of a random number by the voter as a unique identification token, RN. This number facilitates individual verifiability and prevents multiple votes by one voter.

- 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 ($\{\{choice, electionId, RN\}_{blinded}, VT1, electionId, voter\text{-}pub\}_{val\text{-}pub}$).

- 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 ($\{\{\{choice, electionId, RN\}_{blinded}\}_{val\text{-}priv}\}_{voter\text{-}pub}$).

- 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 [21, 22] is built by the Administrator randomly from a set of available nodes ($\{\{\{col\}_{N3\text{-}pub}, N3\}_{N2\text{-}pub}, N2\}_{N1\text{-}pub}$) and passed to the Voter. Ni contains the address of node Ni and its public key, Ni-pub. Although three nodes are used in this example the length of the circuit is variable. The innermost node of the circuit is the Collector (Col) and it is encrypted with the public key of N3, the node that precedes it ($\{col\}_{N3\text{-}pub}$). At the next layer the encrypted innermost nodes are encrypted with the public key of N2, which precedes N3 ($\{\{col\}_{N3\text{-}pub}, N3\}_{N2\text{-}pub}$). The first node on the path, N1, corresponds to the outer layer; all the inner nodes, which are its successors are encrypted with its public key ($\{\{\{col\}_{N3\text{-}pub}, N3\}_{N2\text{-}pub}, N2\}_{N1\text{-}pub}$). During transmission only the successor node is known to its predecessor. Hence, 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, and encrypts it with the public key of N1, ($\{V, \{\{\{col\}_{N3\text{-}pub}, N3\}_{N2\text{-}pub}, N2\}_{N1\text{-}pub}\}_{N1\text{-}pub}$). When N1 receives the message it decrypts it, firstly to access the vote V, and secondly to retrieve the encrypted route to determine the next node in the network ($\{\{\{col\}_{N3\text{-}pub}, N3\}_{N2\text{-}pub}\}$, N2). N1 then constructs a message with the vote V and the rest of the route, and encrypts it with the public of its successor, N2 ($\{V,\{\{col\}_{N3\text{-}pub}, N3\}_{N2\text{-}pub}\}_{N2\text{-}pub}$). The procedure of encryption and decryption is repeated at each node until the message reaches the Collector (col).

- Each server contributes to the monitoring of the voting process by logging and signing explicitly its transactions and authenticating messages where

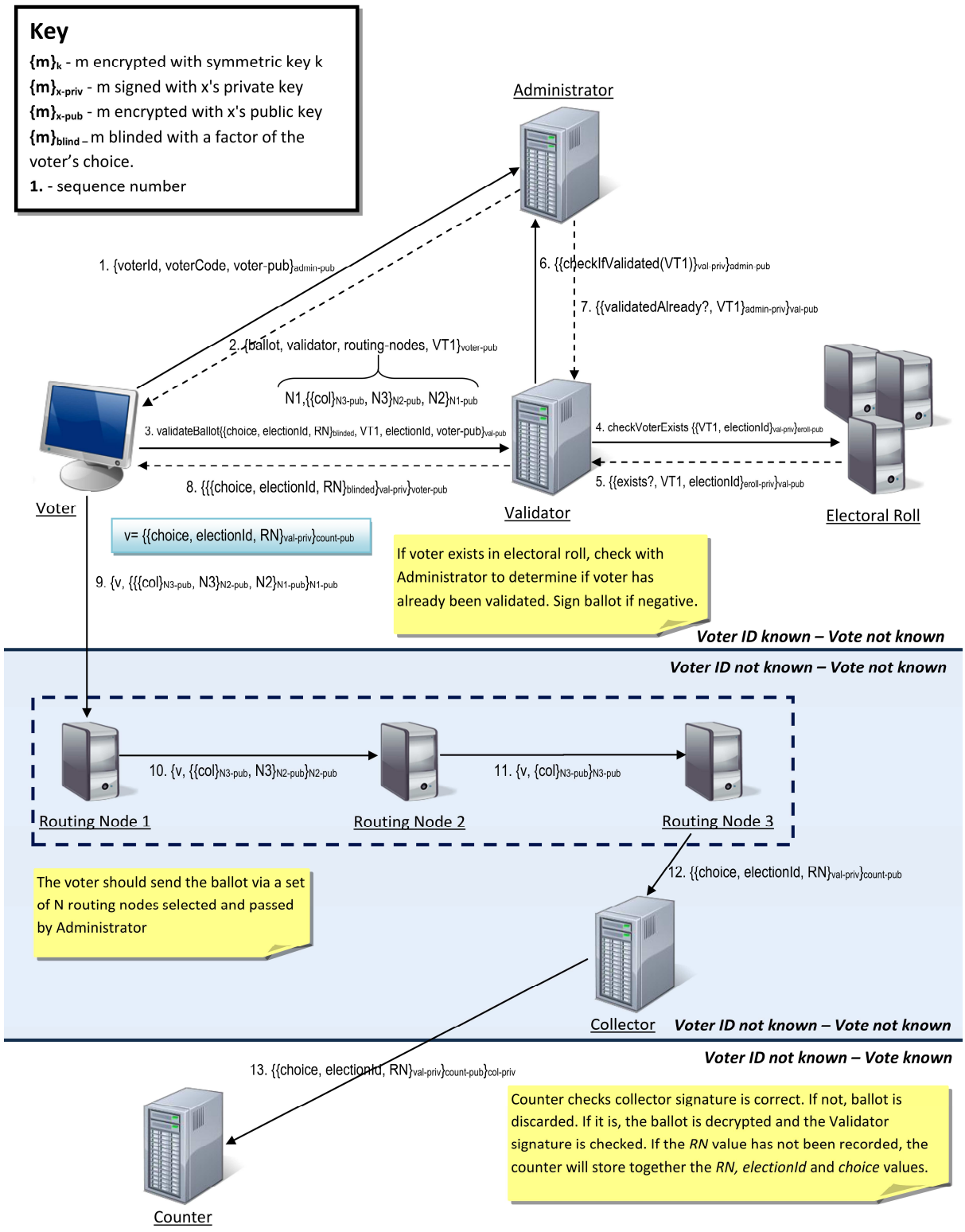

Fig. 1 e-Voting process and architecture

appropriate. Unauthenticated messages are discarded in order to minimise the overload on the network and on the Counter.

5 A service-oriented architecture

The selection of a service-oriented architecture was a key decision in the fulfilment of the e-voting

requirements. One attractive feature of this application is the ability to create aggregate services through dynamic composition**.**

5.1 Web services

As 'self-contained and self-describing applications' Web services offer a number of advantages. Their adherence to well-established standards for Web service description (WSDL), serialisation of messages (SOAP) and Web service indexation (UDDI) underpin their ubiquity and their interoperability. They enable heterogeneous applications to communicate and to be integrated through composition into modular Web services. In addition, they can be deployed over standard Internet technologies and take advantage of the Web infrastructure and protocols [23].

Although the partial statelessness of SOAP/HTTP, as the underlying protocol, is often seen as a drawback in many applications, the intermittent connections of Web services and the regular flushing of state that they initiate make them very suitable for an e-voting application. The absence of state makes them more resilient to failure.

5.2 Architecture

The service-oriented architecture which implements the FOO92 protocol identifies the different stages of the voting process and specifies the roles of the agencies and the entities in the e-voting system. It also represents an instance of the composition of the system from key services. These include:

- Administrator Service: provides a user interface for specifying the election; coordinates the agencies used in an election; serves the Voter Client to the voter and publishes the results when an election ends.
- Electoral roll nodes: they hold voter information. The alias of each voter is mapped to one of the three nodes by a hash function.
- Validator Service: receives the blinded ballot from the Voter Client, using the alias provided by the voter; checks whether the voter exists and whether he or she has not been 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 the collector signature; extracts and records the personal random number (RN) and the vote; adds the vote to the tally.
- Routing Node: receives a ballot either directly from a Voter Client or via a routing node; decrypts the routing path and determines the following node in the path.
- Voter Client: an applet used for casting a vote.

The Administrator service

The Administrator is the most important service in the system. It is the trusted election authority that initiates and coordinates elections. Conceptually it includes seven key components (**Fig.** 2):

- Administrator User Interface: the administrator provides a web-based User Interface (UI) for the Election Official to view the status of agencies, specify elections, view the agencies in use by an election, monitor the election and view the results.
- Voter Client Access Service: this component provides an interface for the voter client to interact with the Administrator service and obtain the list of the candidates and of the agencies for validating and submitting the completed ballot.
- Public User Interface: the voter UI provides a simple Web application to access all the public functions of the system. This includes access to the voter client applet, checking if their vote has been recorded and how it was recorded, and viewing the results of an election.
- Check voter status service: this provides an interface for the Validator service to check if a voter exists, if the voter has voted or to mark the voter as voted if applicable.
- Agency Monitoring: this monitors the status of the agencies in the system. If an agency is in use for an election and becomes unavailable this component selects another suitable one and allocates it to the election (**Fig.** 3).
- Election coordination: this monitors the list of elections in the system and starts and ends them as appropriate.
- Persistence Layer: the persistence layer contains a set of entities which represent the database model. Details of elections, election results, electoral rolls, log messages and agencies used are stored in the database.

5.3 Web Service generation and composition

Web services are implemented within the JAX-WS framework which generates a Web service stub for a service and publishes its WSDL file. This WSDL file is used by applications that consume the Web service to create clients. WSDL files are created for all the services and for different tasks.

The composition process is controlled by the Administrator. 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. 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 of the location of the electoral roll nodes and their public keys:

```
<dhtUpdate>
  <agency id="0" pubKey="1ed$f43fv3s">
    http://eroll5.vote.council.gov.uk</agency>
  <agency id="1" pubKey="3d34v3shbdf">
http://eroll8.vote.council.gov.uk</agency>
  <agency id="2" pubKey="03DX3tfxzy6">
    http://eroll4.vote.council.gov.uk</agency>
  <agency id="3" pubKey="f36hbtgb88r">
    http://eroll0.vote.council.gov.uk</agency>
</dhtUpdate>
```

Fig. 4 presents an outline of the methods that contribute to the composition process. The Collector and the 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.

A key feature of the system is its Just-in-time (JIT) configurability. The JIT strategy is implemented by the dynamic composition of the servers and by the dynamic provision of routing paths.

5.4 Cryptography

The Secure Socket Layer (SSL) was deemed unsuitable for securing 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, 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, which is used to encrypt the message content. 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 so that it can decrypt and access the content of the message. Public keys are distributed to agencies when they are setup as X.509 certificates stored in the key store. A version of an encrypted XML message is shown below:

```
<encrypted-message>
   <sym-key>
      esf234tr4g4t23fgg5y6
   </sym-key>
   <encrypted-content>
      f4wrt3rt5egbdbdfbvt5Y2r3vevsf435gd
   </encrypted-content>
</encrypted-message>
```

5.5 Implementation issues

Java was chosen as the programming language for the implementation of the system because of its suitability for Web development and the availability of libraries for Web services and cryptography. All the application logic was written in EJBs with Glass Fish 3.0 as the application container. EJBs provide many transparent services such as transactions, security, and pooling and thread safety.

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 conformance of SOREV to e-voting requirements, and with the level of robustness it provides at protocol level and at system level. The role of some architectural elements in enhancing robustness is also considered.

6.1 Robustness at protocol level

This form of robustness is assessed in terms of integrity, privacy, verifiability and recoverability.

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.

```
startElection(election) {
        allocateElectoralRollAgencies(election);
        populateElectoralRollAgencies(election);
        allocateAgency("validator", election);
        allocateAgency("counter", election);
        allocateAgency("collector", election);
}
//Allocate an agency to an election e.g. validator, collector, counter.
allocateAgency(agencyType, election, electoralRollNodeId = 0) {
        //Randomly select  one of the available agencies.
        agencyToAllocate = getRandomAgencyOfType(agencyType);
        //Inform agency it has been selected, distribute addresses and keys to agencies
        initialiseAgency(agencyToAllocate, election, electoralRollNodeId);
        recordAllocation(agency, election);
}
allocateElectoralRollAgencies(election) {
       numElectoralRollAgenciesRequired = election.getPrefferedNumberOfElectoralAgencies();
       while (numElectoralRollAgenciesRequired > 0) {
                //select and initialise the node
                allocateNode("eroll", election, electoralRollNodeId);
                numErollAgenciesRequired--;
                electoralRollNodeId++;  }
}
populateElectoralRollNodes(election) {
        for (all voters of election) {
                 //Determine the electoral roll agency to add the voter to.
                   electoralRollAgency = calculateErollAgencyToUse(voter.voterToken1);
                //Add the voter to that agency.
                 ElectoralServiceStub.addVoter(electoralRollAgency, voter.voterToken1);  }
}
monitorAgencies() {
        for (all agencies) {
                If (checkAgencyAlive && !agencySetToAlive)  { agency.setAlive(true);}
                else if (checkAgencyAlive && agencySetToAlive) {//do nothing }
                else if (!checkAgencyAlive && agencySetToAlive) {
                        //this agency has gone down, check if in use.
                        agency.setAlive(false);
                        handleAgencyNotAvailable(agency) }
                 else if (!checkAgencyAlive && !agencySetToAlive) { //do nothing } }
}
void endElection(Election election) {
        notifyNodesThatElectionHasEnded(election);
        //counter will no longer ballots and returns the results of the election.
}
```

Fig. 4 Composition methods

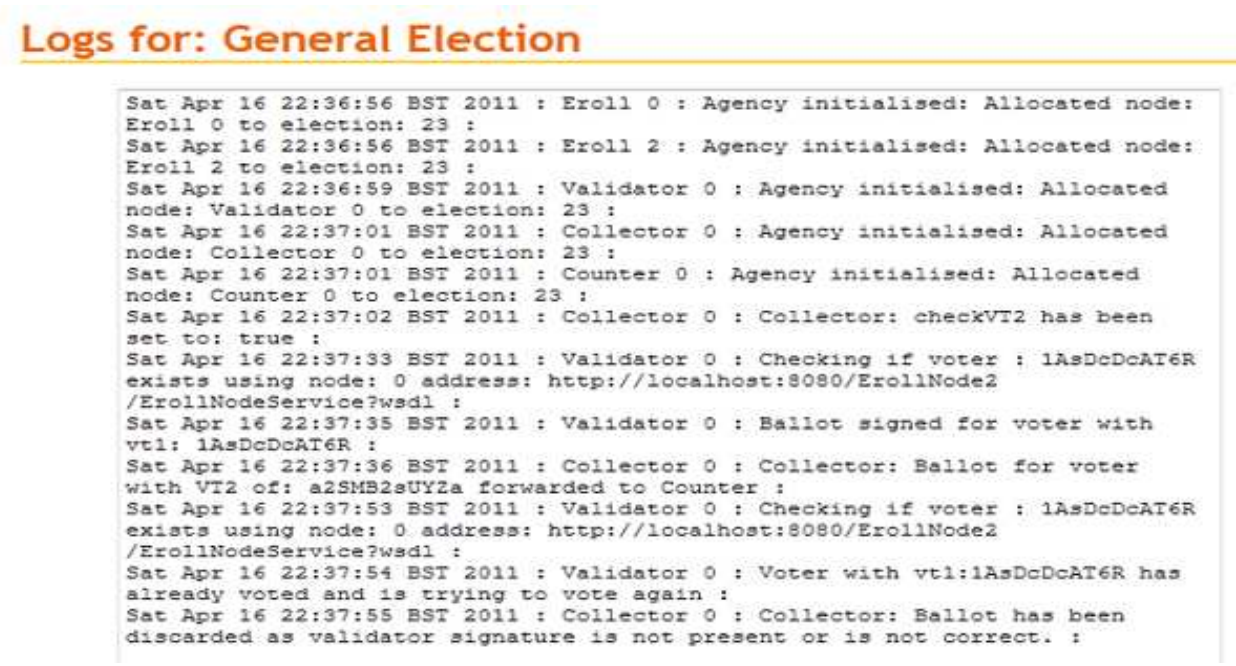

Fig. 5 Server logs

It would be difficult for a Validator to vote on behalf of a voter who abstained. The use of aliases [24] 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 tally of the votes. The provision of multiple Validators would reinforce this security constraint. 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 also 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 (RN) in the Counter allows for votes to be computed accurately and to prevent multiple votes by voters. With the storage of the personal random number with its corresponding vote the replaying of messages is made idempotent and voters are not able to cast two votes.

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. Although it is possible for a routing node to replace a vote by another one, this can only be done with the collusion of the Validator. This will eventually be detected by voters. Votes are only accepted by the Counter if they are sent and signed by the Collector. The injection of spurious votes by entities outside the e-voting system is made difficult by the dynamic generation of the network and therefore its lack of predictability. Whilst existing measures can deter illegal practices, they are ineffective against wholesale collusion between the election authorities.

The integrity of the system is also enhanced by the logs of server transactions and the monitoring of server activity by the Administrator and the Voter (**Fig**. 5). 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 design features contribute to robustness.

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 schemes. Privacy requirements and anonymity in particular, are also supported by architectural features such as onion routing and dynamic routing allocation. The onion routing approach was considered more suitable than mix nets [3] thanks to its impact at the two levels. Mix nets are concerned mainly with untraceability and operate at protocol level.

An additional feature of this implementation of onion routing is that the public keys of the routing nodes do not have to be published, since the chain is created by the Administrator. A node knows only the address and the public key of its successor. Privacy is also ensured by enforcing one-way communications, especially in the last two virtual spaces of the voting process. Privacy

Fig. 6 Individual verifiability

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 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 [18]. This is relevant 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; the focus is instead on verifiability. This characteristic is common to many implementations of the FOO92 protocol such as Sensus [5], SEAS [15] and REVS [16].

The focus of the proposed system was deliberately on the processing of votes and their recording in a robust manner. It is therefore assumed that the vote is cast as intended without support for coercion resistance. It is worth mentioning however that in some situations coercion-resistance may be an important aspect of robustness at protocol level. Civitas implements coercion resistance with the use of fake credentials [17]. Recent elections have confirmed the critical importance of interfacing, one aspect of e-voting to which some researchers had already drawn attention [25].

Verifiability

Individual verifiability and universal verifiability are supported by the accurate recording of the votes and their subsequent publishing without compromising privacy. While individual verifiability can be performed immediately after a vote is cast, for the sake of fairness universal verifiability is only possible after the end of the election. Verifiability can be conducted by the stakeholders in general and the voter in particular by using their personal random number (**Fig.** 6)**.**

Through verifiability and auditing voters can contribute significantly to the robustness of the system at protocol level. This approach is strongly advocated in Helios. Should voters challenge the accuracy of their recorded vote they would have to produce the ballot and the random number signed by the Validator.

Recoverability

The provision for recoverability and its implementation may require the maintenance of state in different servers and at different stages of the voting process. This requirement may not conform to the proposed approach, which is characterised by minimal statefulness in order to safeguard privacy.

In SOREV recoverability is not supported explicitly. Instead a pre-emptive approach was adopted which preserves the integrity and the privacy of the system. Regular and reliable backup is performed on servers that record votes. The state that DHT-based servers hold can be reconstructed without loss of information since it is read-only. Recoverability is facilitated by the dynamic allocation of servers if failures occur.

Recoverability is made possible by the facility that voters have, through verifiability, to check the status of their votes. They can resend their ballot if it has not been recorded.

6.2 Robustness at system level

Robustness at system level will be considered in terms of resilience, scalability, flexibility and cost. More emphasis will be put on resilience as it is the most critical component of robustness.

Resilience

Voter intervention in an e-voting system 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 mitigated by the dynamic generation of the routing path, the absence of obvious patterns in the composition of the system and the verification process, as well as the use aliases. Furthermore, only legitimate voters are given a path to the servers and each voter is given access to the first routing node only.

Fig. 7 Server availability

Denial-of-service attacks can take different forms and many countermeasures were proposed to deal with these attacks at different levels. In some systems filters on the network are used for detecting and blocking DOS attacks [26]. At 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, its flexible configuration with onion routing, 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 mounting concerted DOS attacks. This task is supported by the active monitoring of the servers. The ability to replace defective or rogues servers by new ones can also limit the impact of the DOS attacks (**Fig.** 7).

The dynamic composition of the servers and the provision of dynamic routes can be considered as 'temporal distribution' when opposed to the 'spatial distribution' promoted by REVS and Prêt à voter. Different routes can be active at the same time. The JIT approach enhances reliability as only working and available servers are selected; in addition, the monitoring process helps with the detection of faulty servers and their replacement if required. Furthermore, at a lower level, the chain of routing nodes 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, such as the Validator, routing nodes and the Collector. The distributed electoral rolls (e-rolls) hold the list of aliases of voters, which is read-only and can be restored without loss of information. The Administrator and the Counter maintain however the state that underpins the functionality and the viability of the system. Strong back-up is required in the case of these stateful servers.

Scalability and flexibility

Scalability is primarily catered for by the dynamic configuration of the system. Large volumes of traffic can be managed by redirecting messages adaptively through appropriate paths to servers in order to distribute load, avoid congestion and increase throughput. This function is more effective when combined with server monitoring.

Enhancement to the distributed system can be achieved by the dynamic inclusion of new servers. For example a number of validators, collectors and counters can be integrated dynamically to improve processing or to replace defective servers. Scalability is also implicit in the distribution of state across a set of peers.

Flexibility is manifest at different stages of the lifetime of the system. At initiation, the dynamic composition of the system allows for various configurations to be deployed. In the processing of the votes, the ability to dynamically generate the routing path and the selection of servers to suit environmental conditions is another facet of the flexibility of the system. This also extends to recovery from failure.

The architectural and technological features that sustain the resilience of the system are also key factors in the support for scalability and flexibility. This follows from the adoption of Web services as a versatile technology. With their affinity for interoperability they combine seamlessly client-server and P2P models, and guide the configuration of the loosely-coupled system.

Cost

The expectation of an acceptable level of resilience and the provision for scalability and flexibility in e-voting systems depend on the resources invested in the system and on their cost. The performance of the system can be affected, for example, by the large number of servers that must be maintained and by the creation and distribution of public keys. Some systems opt for a minimal set of servers [15], while others attempt to

satisfy e-voting requirements through multiple and redundant servers [16, 10]. In some cases the robustness of a system depends on the replication of mix nets and on the use of a quorum policy [27], which incurs yet higher performance costs.

In SOREV the overheads are mainly associated with the availability of multiple servers and the processing of the routing path. The 'validation virtual space' is the context of intensive two-way inter-server communications, while the 'transmission space' is marked by heavy encryption and decryption.

6.3 Architectural impact

This section offers a brief assessment of the impact of some architectural features on robustness at protocol level and at system level.

Virtual spaces

Although most e-voting systems operate implicitly within two virtual spaces only, three distinct phases are identified in this system: validation, transmission and recording. While transmission and recording are relatively secure, validation involves many servers and two-way communications. The level of activity and the patterns of behaviour it supports may expose the servers to malicious attacks. The distribution of state and tasks and the random selection of the servers form a significant part of the measures against potential attacks.

Clear identification of roles of the servers and active monitoring of server behaviour are features that help pre-empt single points of failure and deter collusion. The use of aliases and the dynamic generation of the routing path contribute to voter privacy and the security of the distributed system.

A service-oriented architecture

The adoption of Web services allows for a considerable overlap between resilience, scalability and flexibility. This is also enhanced by the stateless nature of the combination of the HTTP and SOAP protocols and their support for ephemeral state information. One significant feature against denial of service attacks is the limited awareness that the servers have of each other. Additionally, one-way messages can hinder the identification of the source of a message through traffic analysis.

As composition is initiated by the Administrator and involves a number of trusted Web services that possess security capabilities and are subject to security constraints, it can be stated that the composition process is inherently security conscious [28]. Moreover, the resilience of the system is enhanced by the loosely-coupled configuration of Web services.

Architectural components

The merits of the architectural and technological features of the system have been outlined in the previous sections. A more focused assessment of their impact on robustness at the levels is given in **Fig.** 8**.** The elements of interest include the service-oriented architecture, JIT composition, e-roll nodes, onion routing and one-way communication. The table identifies the core components of robustness that were affected. At protocol level it is the integrity and privacy, whereas at system level it is mainly resilience and scalability and flexibility. Integrity and privacy are both supported by strong encryption.

6.4 Comparative evaluation

A comparative evaluation with other, albeit older, implementations of the FOO92 protocol will shed some light on the different forms of robustness and its distinctive features. Unlike more recent implementations of e-voting schemes which are mainly concerned with protocol level, systems such as Sensus and REVS have the merit of presenting concrete implementations that implicitly or explicitly address robustness at system level as well.

At protocol level the comparative evaluation (**Fig.** 9) indicates that no system satisfies fully the criterion of recoverability. In many respects Sensus is weaker than REVS and SOREV. In addition to the lack of universal verifiability its support for integrity is restricted and anonymity is not guaranteed. REVS and SOREV present similar capabilities but differ in the way they implement integrity. REVS relies mainly on an explicit quorum policy while SOREV combine aliases, distributed state, intermediate servers and dynamic routing to achieve integrity.

At system level (**Fig**.10) both REVS and SOREV have benefited from the experience of Sensus and satisfy most of the e-voting requirements; they display a number of advanced features. Sensus was however designed with minimal resources and with multi-function servers; as a result its overheads are lower.

Although REVS and SOREV are close in many ways, the fundamental difference between them lies in the way they deal with resilience. REVS opts for the redundant replication of servers with alternative routing to support a quorum policy. Moreover the need to facilitate 'resumability' requires the maintenance of state across the whole system. SOREV, on the other hand, favours the dynamic allocation and configuration of servers and the dynamic route generation. To some extent the 'spatial distribution' of resources in REVS is equivalent to their 'temporal distribution' in SOREV. This difference has implications for scalability, flexibility and cost. Although both provide equivalent support for scalability SOREV offers more flexibility at system level than REVS and makes full use of its resources, a feature that helps minimise cost.

In REVS the cost is mainly due to the overheads of the quorum policy; in SOREV it is the route generation that

requires heavy processing. It can be concluded from the two tables that REVS performs better at protocol level, whereas SOREV satisfies better the system level criteria.

This comparison may present Sensus in a relatively poor light. The system has the merit of being one the first realistic implementations of the FOO92 protocol. It served as a model for many e-voting schemes. Some of its limitations are mainly due to the restricted number of servers. In contrast to previous systems, some recent e-voting systems such as Civitas, Prêt à voter and Helios appear to be more concerned with the protocol level and with a stronger form of verification. Despite their concern with integrity and privacy, with the exception of Civitas, coercion resistance is still not supported in many e-voting systems.

6.5 Contribution

The review of e-voting systems has highlighted an overwhelming concern with protocol design and conformance to e-voting requirements. In most of these systems robustness at system level is not dealt with explicitly. It is often assumed that techniques for addressing robustness can be subsequently grafted onto the system.

The main contribution of this work lies in the explicit approach to robustness in e-voting systems. This involved firstly, a distinction between two types of robustness and secondly, the design of a service-oriented architecture which integrates appropriate technologies in order to address the two forms of robustness simultaneously.

The design of the underlying distributed system was guided by a number of concepts [8]:

- distributed trust concepts, separation of concerns and processing to prevent collusion.
- restriction of access to state information to deter cheating.
- distribution of e-roll state, use of aliases and unique identifiers to enhance the privacy of voters.
- establishment of stateless servers to enhance reliability and recovery.
- dynamic assignment of routes and servers to counter malicious behaviour and facilitate efficient system behaviour.

Although the focus of the work is on robustness the proposed architecture provides full support for the e-voting process and satisfies all the e-voting requirements with the exception of coercion resistance. The identification of three virtual spaces played a significant part in achieving this conformance.

7 Further work

The approach to the development of e-voting systems offers scope for improvement and extension. Areas for further are sketched below.

7.1 Enhancement with BPEL

The Web Services composition process is ad hoc. A more disciplined approach can be supported by enabling WS-BPEL to preside over the management of Web services. WS-BPEL offers better transaction handling and can be used to orchestrate a set of web services to perform a specific task.

WS-BPEL could be used to formalise the description of the interactions between the different Web services in the system. In addition to starting and ending an election, this would facilitate the process of monitoring the behaviour of agencies in the system. If a change of status was found the exception handlers would be called accordingly.

7.2 Server replication

The Administrator performs many functions and holds information critical to the whole process. This centralisation can be detrimental to the integrity of the system. A more robust and secure approach would involve the generation of the codes and aliases by a separate election authority followed by their distribution to different servers. This would prevent the Administrator from, for example, fabricating false identities and voting on their behalf or colluding with other agencies.

Similarly, a single Counter is vulnerable to attacks and may be a single point of failure. It is possible to improve the resilience of the system by specifying different Counters through the Collector or a set of Collectors. The final tally will be gathered from all the Counters. Alternatively two counters can be provided that receive similar messages from the Collector. This scheme ensures resilience, verification and accuracy through mirroring. This will however incur some performance overheads.

7.3 Onion routing

Onion routing has the advantage of promoting decentralisation and distribution. This method of communication could be generalised to the exchange of messages between servers. It can present a more effective defence against denial of service attacks and collusion, since the routing nodes are arbitrary selected. Servers need not be aware of the source of a message as long as the signature is valid. The introduction of further redundancy into the system may be costly and may affect adversely its reliability.

7.4 Mobility

Many are advocating the use of mobile devices in e-voting systems [29]. It is argued that the ability of voters to vote from their mobile devices would lead to greater flexibility and would encourage greater

participation in elections. Although this enhancement would meet the mobility requirement, the introduction of mobile devices raises a number of issues. Mobile networks are notoriously difficult to manage; the ad hoc and intermittent interventions of mobile devices pose serious issues of authentication and security. It is however the current limitations of these devices that could be the main obstacle to their integration into e-voting systems [30]. In addition to computational and memory constraints battery restrictions can lead to discontinuity in processing and loss of votes.

8 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 is characterised by a distinction between robustness at protocol level and robustness at system level, and the identification of technologies and their integration into a service-oriented architecture in order to address the two forms of robustness simultaneously. With its emphasis on the distribution of state, the decentralisation of tasks, the JIT routing configuration, the use of aliases and the active monitoring of server activity, the proposed approach aims at promoting the design of robust e-voting systems. This is supported by the implementation of the servers as Web Services and their dynamic composition. This perspective on robustness offers scope for wider decentralisation, scalability and flexibility, 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.

9 References

1. Kremer S., Ryan M. D. and Smyth B. Election verifiability in electronic voting protocols. *In Proceedings of the fifteenth European Symposium on Research in Computer Security (ESORICS'10). LNCS, Springer, volume 6345*, pp389-404, 2010.

2. Chaum D., Untraceable electronic mail, return addresses, and digital pseudonyms, *Communications of the ACM 24 (1981) (2)*, pp84–88.

3. Sampigethaya K. and Poovendran, R. A Survey on Mix Networks and Their Secure Applications, *Proceedings of the IEEE, Volume: 94, Issue 12*, December 2006, pp2142–2181.

4. Syverson P.F., Goldschlag D.M. and Reed M.G. Anonymous connections and onion routing, *IEEE symposium on security and privacy, IEEE (1997*), pp44–54.

5. Cranor L. and Cytron R.K. Sensus: a security-conscious electronic polling system for the internet, *Proceedings of HICSS'97, IEEE (1997)*, pp561–570.

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

7. Volkamer, M. and Grimm, R. Determine the Resilience of Evaluated Internet Voting Systems. *First International Workshop on Requirements Engineering for e-Voting Systems (RE-VOTE)*, Atlanta, USA, August 2010, pp47-54.

8. Weldemariam K., Volkamer M. And Villafiorita A. "e-voting: What we Learned, Where We are Going To". *Proceedings of the Sixth International Workshop on Frontiers in Availability, Reliability and Security, (FARES 2011)*, Vienna, Austria, August 2011.

9. Schneier B., *Applied Cryptography*, John Wiley, 1996.

10. Ryan P.Y.A., Bismark D., Heather, J. Schneider S. and Zhe X. The Prêt à voter Verifiable Election System, *IEEE Transactions on Information Forensics and Security - Special issue on electronic voting, Volume 4, Issue 4*, December 2009, pp662-673.

11. 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.

12. Chaum D. "Blind Signatures", *Crypto 82*, 1983.

13. Ansari N., Sakarindr P., Haghani E., Zhang C., Jain A.K., and Shi Y.Q., Evaluating Electronic Voting Systems Equipped with Voter-Verified Paper Records, *IEEE Security & Privacy*, May/June 2008, pp30-39.

14. Scott Jackson, A Multidisciplinary Framework For Resilience To Disasters And Disruptions. *Journal of Integrated Design & Process Science, Volume 11 Issue 2*, April 2007.

15. Baiardi F., Falleni A., Granchi R., Martinelli F., Petrocchi M. and Vaccarelli A. SEAS, a secure e-voting protocol: Design and implementation. *Computers & Security*, 2005, pp642-652.

16. Joaquim R., Z´uquete A., and Ferreira P. REVS – A Robust Electronic Voting System. *IADIS International Journal of WWW/Internet, 1(2)*, December 2003.

17. Clarkson M. R., Chong S., and Myers A. C. Civitas: Toward a secure voting system. *IEEE Symposium on Security and Privacy*, Oakland, USA, 2008, pp54–368.

18. Adida B., Helios: Web-based open-audit voting. *Proceedings of the 17th USENIX Security Symposium (Security '08)*, San Jose, USA, July-August 2008.

19. Carl G., Kesidis G., Brooks R.R. and Rai S. Denial-of-Service Attack-Detection Techniques, *IEEE Internet Computing, Volume 10, Issue 1*, 2006, pp82-89.

20. 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.

21. Dingledine R., Mathewson N. and Syverson P., Tor: The Second-Generation Onion Router, *Proceedings of the 13th USENIX Security Symposium*, August 2004.

22. Camenisch, J. and Lysyanskaya, A. A Formal Treatment of Onion Routing. *Proceedings of CRYPTO'2005, Lecture Notes in Computer Science, Vol. 3621*, November 2005, pp169-187.

23. Curbera F., Duftler M., Khalaf R., Nagy W., Mukhi N., Weerawarana S., Unravelling the Web Services Web - An Introduction to SOAP, WSDL, and UDDI, *IEEE Internet Computing, 3 (4)*, 2002.

24. Langer L., Schmidt A., Buchmann J. and Volkamer M. A. Taxonomy Refining the Security for Electronic Voting: Analysing Helios as a proof of Concept. *2010 International Conference on Availability, Reliability and Security*, Krakow, Poland, February, 2010.

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

26. .Abdelsayed, S., Glimsholt, D., Leckie, C., Ryan S. and Shami S. An efficient filter for denial-of-service bandwidth attacks, *IEEE Global Telecommunications Conference, (GLOBECOM '03), Volume: 3*, 2003, pp353-1357.

27. Jakobsson M., Juels A. and Rivest R.L. Making Mix Nets Robust for Electronic Voting by Randomized Partial Checking. *USENIX Security Symposium*, 2002, pp339-353.

28. Carminati B., Ferrari E and Hung P.C.K, Security Conscious Web Service Composition., *2006 IEEE International Conference on Web Services (ICWS 2006)*, September 2006, Chicago, USA, pp489- 496.

29. Campanelli, S., Falleni, A., Martinelli, F., Petrocchi, M., Vaccarelli A. A Mobile Implementation and Formal Verification of an e-Voting System. *Internet and Web Applications and Services*, (*ICIW '08*), 2008, pp476-481.

30. Ashraf K., Anane R. and Bordbar B. File Management in a Mobile DHT-based P2P Environment. *The 26th IEEE International Conference on Advanced Information Networking and Applications (AINA-2012)*, Fukuoka, Japan, March 2012.

Architectural feature	Protocol level	System level
Service-oriented architecture and JIT composition	Integrity	Resilience Scalability Flexibility
e-roll node	Integrity Privacy	Resilience Scalability
Onion routing	Integrity Privacy	Resilience Scalability Flexibility
One-way routing	Privacy	Resilience Scalability

Fig. 8 Architectural impact

Robustness at Protocol level			
	Sensus	**REVS**	**SOREV**
integrity	Potential cheating by Validator (resolved in SEAS) Potential collusion between servers	Collusion resistance through quorum policy Votes cannot be forged because of multiple signatures and quorum policy Can be sensitive to DOS attacks because of maintenance of state at different stages	Collusion resistance through distribution of tasks Indirect access to e-roll data with the use of aliases Use of transaction logs Most operations are idempotent
privacy	Strong encryption of ballot (secret vote) Two undifferentiated virtual states Anonymity not guaranteed (assumed) Receipt-based (two-way communication) Coercion resistance not supported	Two virtual spaces Communications signed and encrypted Receipt-free system Unique identification of ballot Coercion resistance not supported	Strong encryption of ballot Three virtual spaces Signed transactions Receipt-free system Use of random UUID (RN) for personal verification Coercion resistance not supported
verifiability	Individual verifiability supported Universal verifiability not supported	Individual verifiability supported Universal verifiability supported	Individual verifiability supported Universal verifiability supported
recoverability	Not addressed explicitly Inaccurate votes can be detected by voter; final tally can be corrected at the end of the election if required	Not addressed explicitly Pre-emptive approach to cheating and failure through replication of servers, quorum policy and distribution of state	Not addressed explicitly Can be achieved by a combination of Voter and Counter records and tallies; this might compromise privacy Pre-emptive approach to deter and minimise effect of malicious behaviour, through distribution of state and tasks

Fig. 9 Protocol level comparison

Robustness at System level			
	Sensus	**REVS**	**SOREV**
resilience	Existence of anonymous channel assumed but not implemented Server can act as single point of failure	Achieved through distributed loosely-coupled servers Replication of servers to ensure resilience to DOS attacks Alternative servers/paths can be used	Anonymous channel supported through onion routing Dynamic route generation One-way communication in the last two virtual spaces System monitoring The most vulnerable part is the Validator; can be subject to DOS attacks (known to voters) Dynamic replacement of faulty servers
scalability	Fixed set of servers System can however be augmented by additional servers	Scope for expansion in the replication of servers Supported by parallel selection and operation of servers Many elections can be run at the same time	Supported by JIT dynamic configuration of servers and routing path Facilitated by Web Services Distribution of state and tasks Support for multiple simultaneous elections
flexibility	Static configuration Tightly coupled systems Flexibility of ballot format Unix-based system	Static configuration involving a number of servers Flexible replication of servers RMI over SSL communication (need for Java Virtual Machine)	Dynamic Web Services composition Dynamic route generation Flexible configuration Loosely coupled systems Dynamic reallocation of servers after failure or cheating SOAP-based communication
cost	Three types of servers Stateful servers Minimal number of messages Efficient use of resources and processing	Five types of servers Replication of servers Quorum policy Stateful client and servers (resumability) Heavy communication between servers at all levels Use of multiple servers performing the same task, some of which many may be idle Heavy asymmetric encryption	Six types of servers Dynamic allocation and processing of routing path Multiple e-roll servers Two-way messages in the first virtual space (validation) One-way messages in the transmission and recording virtual spaces Use of open source technology Heavy asymmetric encryption

Fig. 10 System level comparison