e-vota

Alice, Melusi, (& Jeff)

MISG 2016 student workshop

Abstract. The purpose of this project has been to consider desirable properties of a distributed 'electronic voting system' and to provide a high-level design ensuring them. Learning objectives have been: (a) how to express non-functional requirements (b) how to describe a distributed system (c) how to reason about distributed behaviour.

1 Introduction

Assumptions:
(a) The electoral roll(voters register) is up-to-date implying that no addition of new voter or deletion of an existing voter.

2 Requirements

(a) Voter Authentication
(b) Confidentiality
(c) Accountability
(d) No double voting

3 The class *e-vota*

A system is viewed as a class (aka object, abstract data type, module, ...): having states, initial states and operations (which may take input, update state and give output). The class is described using the Z notation.

The Z class has these generic types:
(i) *Strings*, of voters
(ii) *Info*, of voters' info (Demographic, geographic, ...)
(iii) *Candidates*, of candidates (Candidates $\neq \varnothing$)
(iv) *Time*, of the voting period
(v) $Deadline \in Time$
(vi) *vp* (voting protocol) is the method that computes the rankings from the votes cast by voters that is

$$vp : multiset - range(votescast) \nrightarrow rankings$$

3.1 System state

The state of the system contains the set of voters and the votes cast so far.

State

$$voters : \mathbb{P} Strings$$
$$votescast : voters \pfun Rankings(Candidates)$$

$$voters \neq \varnothing$$

3.2 Initially

sets Initially no votes have been cast.

Init

$$State$$

$$votescast = \varnothing$$

The set of voters remains unchanged throughout the operations.

3.3 System operations

This operation checks for the validity of both the voter and the vote at one go.

ValidateVoter&Vote

$$\Xi State$$
$$voterid?, voterid! : String$$
$$vote?, vote! : SeqCandidates$$
$$valid! : \mathbb{B}$$

$$valid! = (voterid? \in voters) \land (vote? \in Rankings(Candidates))$$
$$voterid! = voterid?$$
$$vote! = vote?$$

Depending on the result from the above operation, the vote can either be counted or not.

AddVote

$$\Delta State$$
$$valid? : \mathbb{B}$$
$$vote? : SeqCandidates$$
$$voterid? : String$$
$$votermsg! : \mathbb{B}$$

$$valid? \Rightarrow votescast' = votescast \oplus \{(voterid?, vote?)\}?$$
$$\neg valid? \Rightarrow votescast' = votescast$$
$$votermsg! = valid?$$

The operation of casting a vote is now described in terms of those two smaller operations, using 'piping'. This is simply a specification method, and does not imply that the implementation needs to reflect the two operations individually.

$$CastVote := ValidateVoter \& Vote \quad \gg \quad AddVote.$$

The precondition for *CastVote* holds: it is a total operation.

Announce

$\Xi State$
$t : Time$
$result! = Rankings(Candidates)$

$t = Deadline$
$result! = vp(multiset - range(votescast))$

The precondition for *Announce* is $t = Deadline$.

4 Properties

Establishing the properties.

4.1 No double voting

4.2 Voting Module

System design involves a number of levels from high to lower levels of abstraction. For the voting system, splitting the atomic vote into several operations and actions raises concerns. We show how such concerns can be addressed.
For a set A of voters and each $a : A$, let
k_a be a's public key, $\overline{k_a}$ be a's private key.
Then $k_a \bullet \overline{k_a} = \overline{k_a} \bullet k_a =$ identity.
(Also $k_a \bullet k_b = k_b \bullet k_a$).

4.3 Digital Signature

The voter encrypts his/her vote with $\overline{k_{id}}$ (which no one else is able to do, because $\overline{k_{id}}$ is private). The system then authenticates id's vote by applying k_{id}.
Assumption: for $u \neq id$. $k_u \bullet \overline{k_{id}} \notin Rankings(Candidates)$.
A hacker might do $\overline{k_u} \bullet k_{id} \bullet \overline{k_{id}}$ and claim id's vote as his own. but that's another problem.
This technique ensures voter accountability (given the assumptions of Public Key Encryption).

5 Conclusion

References

1. L. Fouard, M. Duclos and P. Lafourcade. Survey on electronic voting systems. *??*, **?**:?–??, 20??.