2010 International Conference on Availability, Reliability and Security

Formal Specification and Analysis of an e-Voting System

Komminist Weldemariam[1,3]
[1]*Dept. of Eng. and Comp. Science*
University of Trento
Trento, Italy
Email: sisai@fbk.eu

Richard A. Kemmerer[2]
[2]*Dept. of Comp. Science*
University of California
Santa Barbara, USA
Email: kemm@cs.ucsb.edu

Adolfo Villafiorita[3]
[3]*Center for Information Technology,*
Fondazione Bruno Kessler (FBK-irst)
Trento, Italy
Email: adolfo@fbk.eu

***Abstract*—Electronic voting systems are a perfect example of security-critical computing. One of the critical and complex parts of such systems is the voting process, which is responsible for correctly and securely storing intentions and actions of the voters. Unfortunately, recent studies revealed that various e-voting systems show serious specification, design, and implementation flaws.**

The application of formal specification and verification can greatly help to better understand the system requirements of e-voting systems by thoroughly specifying and analyzing the underlying assumptions and the security specific properties.

This paper presents the specification and verification of the electronic voting process for the Election Systems & Software (ES&S) system. We used the ASTRAL language to specify the voting process of ES&S machines and the critical security requirements for the system. Proof obligations that verify that the specified system meets the critical requirements were automatically generated by the ASTRAL Software Development Environment (SDE). The PVS interactive theorem prover was then used to apply the appropriate proof strategies and discharge the proof obligations.

***Keywords*-Electronic Voting Systems, ES&S system, Formal Specification and Verification, Critical Requirements.**

I. Introduction

Electronic voting systems are increasingly replacing the traditional paper-based voting systems. These systems can make the voting process more convenient and may, therefore, lead to improved turnout. Electronic recording and counting of votes could be faster, more accurate, and less labor intensive. However, as witnessed in [1], [2], [3], [4], [5], e-voting systems often share critical failures in their design and implementation, which render their technical and procedural controls insufficient to guarantee trustworthy voting. In California, these studies resulted in the Secretary of State allowing the use of e-voting machines only in special situations and with various changes to the electoral procedures. Several such changes shift the implementation of security requirements from e-voting systems to poll workers. For instance, [6] states that "no poll worker or other person may record the time at which or the order in which voters vote in a polling place". It is quite evident that a new generation of more carefully designed and engineered machines could move various checks currently performed by poll workers back to hardware and software.

However, the success of this new generation of machines depends on our ability to capitalize from the lessons we learned using and analyzing the systems currently deployed. The work we present in this paper is one way in which we can get a better understanding of the strengths and the weaknesses of existing systems and thus lay the foundations for engineering and deploying a new generation of more secure and robust technologies for polling stations.

We do so by presenting the formal specification of the voting process for the Election System & Software (ES&S) voting system as documented in the EVEREST project report [7]. We treated the ES&S voting system as a complex, real-time embedded system, consisting of a direct recording election machine (DRE), a real-time audit log printer (RTAL), a personalized election ballot (PEB), and a Compact Flash Card (CF). We mapped each of these components to ASTRAL [8] process instances, and we also specified critical security requirements to prove the correctness and integrity of the election results. The consistency of the specification was validated using the ASTRAL validation engine, and PVS [9] proof obligations were automatically generated by the ASTRAL Software Development Environment (SDE). When proved, these proof obligations verify that the specified system meets the critical security requirements. The PVS interactive theorem prover was used to apply the appropriate proof strategies and discharge each of the proof obligations.

This paper is structured as follows. The next section presents the motivation for this work. In Section III, we summarize the main components of the ES&S system and the voting process when using this system. The section also presents samples of critical security requirements that the system should meet. The ASTRAL specification of the system and its critical security requirements and the status of the PVS formal verification is presented in Section IV Finally, we discuss related work and draw our conclusions in Sections V and VI.

II. Motivation

Fairness and security of electronic elections depend upon a careful allocation of requirements to the procedures and to the systems used. In fact the correct behavior of the

978-0-7695-3965-2/10 $26.00 © 2010 IEEE
DOI 10.1109/ARES.2010.83

electronic systems can be guaranteed only when they are used according to their operating specifications. This has to be guaranteed by the procedures and the people responsible for executing them. For example, there is no way for an e-voting machine to prohibit the same person from casting multiple ballots, if the poll workers enable the machine for voting multiple times. This behavior can be prevented (or revealed after the election) only by enforcing and verifying the procedures that the poll workers are supposed to follow.

In contrast, there are other fundamental properties that the procedures can only partially assure. In this case, the e-voting systems must guarantee that these properties are satisfied. Using the example we just made, the machine must ensure that a voter can cast at most one vote, given that the poll workers follow the prescribed procedures.

Analyzing such requirements and their allocation is therefore important in two respects. First, it helps to ensure that the systems meet the necessary reliability and dependability goals. Second it helps to better understand how a different allocation of requirements between systems and procedures could improve the overall security of the election process so that we can build the next generation of e-voting machines. However, in order to achieve these goals, we need an approach that allows us to easily experiment and reason about, for example, different allocations of requirements. At the same time, it has to be precise and exhaustive, so that security and dependability consequences of any specific choice are highlighted.

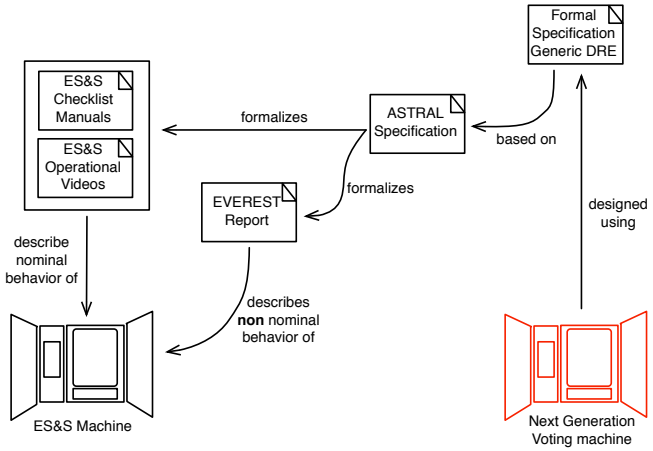

Figure 1. Overview of our approach. Labels nominal and **non** nominal refer to the *actual* and *undesired* behaviors of the system, respectively.

Formal techniques clearly fit both needs, and the goal of this work is to demonstrate how their use can help to ensure fair elections. More specifically, we derive formal specifications along with critical security requirements for one of the currently deployed e-voting systems (i.e., ES&S system) using the ASTRAL language. The specification of the system and the critical requirements are mainly derived from available information sources: the EVEREST report [7], the ES&S election day checklist manual [10], a video[1] that shows how the ES&S system works on election day, and other requirements suggested in the literature (e.g., [11], [12]). Using formal analysis tools, we can assess the strengths and weaknesses of the system. Results or feedback gained from the formal analysis can be a basis for specifying and analyzing generic requirements from which the next generation of voting machines can be built (see also Figure 1).

III. Electronic Voting Systems

A. ES&S Voting System Components

For the purposes of this work (see [7] for a more detailed and complete view) the ES&S voting system is composed of:

- *DRE*: Direct Recording Electronic voting machine, called the iVotronic. It is equipped with a touch-screen where the voter casts his/her votes. The information shown by the touch-screen changes in real-time to match the voter's choices. The iVotronic also stores the audit data.
- *RTAL*: Real-Time Audit Log Printer, which performs the function of a VVPAT (Voter-Verified Paper Audit Trail) for the ES&S system. It produces a paper-based record of the choices selected by the voter. The RTAL is plugged into the DRE and the paper record is viewable by the voter. The voter's choices are under a transparent cover so that they cannot be modified other than through the normal voting process.
- *PEB*: Personalized Electronic Ballot. This is a device used by the poll worker to load a ballot, initialize the next ballot, and collect tabulated data and audit information. Each time a PEB is inserted, its authenticity is checked by the DRE using a four-digit code (election qualification code, EQC), which is assigned prior to election day.
- *CFC*: Compact Flash Card. This device holds files too large to fit in the PEB and also audit data. The card must be present to open and close the polls. At poll closing, the audit data is automatically dumped into the card.

B. Voting Process for a DRE based System

In short, running an election on election day using the ES&S system is as follows (see [7] for further detail). Prior to opening the polls, a poll worker unpacks and sets up the DRE and plugs in the RTAL printer and power cables. Poll workers must also ensure that a properly programmed CF card is installed before powering on the DRE. A Master PEB is inserted into the terminal to load the ballot and later to open the terminal for voting. The same master PEB must

[1] http://www.essvote.com/HTML/voter_outreach/ivotronic_flash.html

be used to close the terminal after the polls have closed. Removing the PEB turns the terminal's current mode to sleep mode.

Once the polls are opened, a poll worker initializes the ballot for a qualified voter by inserting a supervisor PEB, which can be the same Master PEB used to open the polls, into the machine. The terminal mode changes from sleep to poll worker mode, the EQC code of the PEB is checked, and the ballot is initialized, provided that the EQC of the PEB matches with the one the terminal was configured for. The poll worker removes the supervisor PEB and leaves the terminal for the voter.

After the ballot is activated, the machine takes the voter through each contest. DRE machines automatically forbid overvoting, but not undervoting. When a voter selects or cancels a candidate for a particular contest, an appropriate indication is printed on the RTAL record. If the voter selects a candidate, the RTAL record is marked as "Selected" and scrolled out of sight; otherwise, it is marked as "Canceled" and scrolled out of sight. The voter is eventually given the opportunity to review his ballot, and if the voter commits to it (confirms it), it is recorded to local storage. The process continues in this way for all qualified voters.

After the official poll closing time is reached and there is no qualified voter waiting in line, the poll worker inserts the master PEB to collect and store tabulated data, copies of the ballot image, and some other information. Upon closing the terminal, the DRE firmware automatically uploads the audit data onto the CF card. The results tape from the RTAL is also collected. The results tape, CF card, and master PEB from each polling place are then returned to election central.

C. Informal Requirements for the ES&S System

A number of requirements that the ES&S system must satisfy are enumerated in the ES&S system manual [10] and a corresponding video. Below we present a sample of the most important requirements (see [13] for more details).

A correctly functioning DRE must satisfy the following properties.

- *D-Req1:* The DRE must authenticate the PEB using the EQC, and the same master PEB must be used to open and close the terminal;
- *D-Req2:* Display screens presented to the voter must accurately reflect the ballot downloaded from the PEB and the selections made by the voters;
- *D-Req3:* The DRE terminal only allows two valid actions for the voter until he/she reaches the final review (vote summary) screen: (1) select or cancel a candidate on the screen, or (2) move forward or backward through the ballot;
- *D-Req4:* For each valid voter action (i.e., starting to vote, making a select or cancel, and finishing a vote) the DRE must enable the RTAL to record the action on the RTAL tape accordingly;

The RTAL must satisfy the following properties:

- *R-Req1:* The RTAL should scroll up a minimum distance after the summary has printed;
- *R-Req2:* The RTAL must update the paper tape after the voter pushes the start button, makes a choice (select or cancel), confirms the vote, or when a poll worker rejects the ballot of a fleeing voter.

The PEB must satisfy the following properties:

- *P-Req1* The election-specific secret code (EQC), which is a 32-bit (4 digit) code, must be present on a PEB and must always match with the one stored inside the DRE; otherwise, the PEB should be rejected by the DRE terminal whenever the poll worker attempts to insert it;
- *P-Req2* At the end of the election, the copy of the ballot images downloaded from the DRE must be the same as the ballot images that were loaded into the DRE prior to starting the election.

The CF card should satisfy the following property:

- *C-Req* The poll closing procedures must copy the audit information (such as the event log) accumulated in the local storage to the CF card.

The following global properties must be ensured by the system components all together:

- *G-Req:* No discrepancy should be observed among the following: (1) the individual cast ballot records (or ballot images) recorded by the machines; (2) the summary tape generated on Election Day at the close of polls on individual machines; (3) the totals that were accumulated and reported by the DRE and RTAL.

The above requirements (and those that are not listed in this paper) are converted into ASTRAL invariants, schedules, and constraints.

IV. Specification of the ES&S Voting System

The complete ASTRAL specification of the ES&S is approximately 30 pages long. In this section, therefore, we are able to present only a sampling of the specification to provide a flavor of the work (see [13] for detailed discussion of the specification)

A. ASTRAL Specification of the ES&S

We specify each component of the electronic voting process as an ASTRAL process instance. Four process types are declared in the global specification of the ASTRAL model of the ES&S. Below is an example of a process declaration:

```
PROCESSES
 the_DRE: array [1..Number_Of_DRE]
     of DRE_Process [...],
```

We defined user defined types and constants to represent useful concerns about the ES&S system inputs and outputs, like in the following snippet specification:

```
TYPE
 PrintValue/*Unspecified type*/,
 Candidates IS SET OF Candidate,
 Race_Candidate_Pair IS STRUCTURE OF
    (Contest: Race, Nominees: Candidates ),
 Ballot IS SET OF Race_Candidate_Pair,
 Decision IS ( Selected, Canceled),
CONSTANT
  Plugged_In_RTAL( DRE_ID ): RTAL_ID,
  Make_Print_VoteEntry(Name, Title,
                  Decision): PrintValue,
[...]
```

1) Modeling the DRE Process: The ES&S DRE device is modeled by the process type `DRE_Process`. To model permissible operations on the DRE machine, we need to capture the phases of the election and the various modes of the terminal during election day. We use the variables `Which_Phase, Terminal_Mode`, and `DRE_State` of types `Election_Phase, Mode` and `Terminal_State`, respectively. These indicate, respectively, the phase of the election (pre-voting, during-voting, and post-voting), the terminal mode (poll worker, voter, sleep, or chirping), and the state of the poll (opening, opened, closing, or closed).

When a voter casts a vote, s/he is actually interacting with the system by navigating from one screen to another using an appropriate button. We modeled such interaction by assigning an integer number to each screen shown to the voter and by defining specification functions that take as input a screen number and return the information to be displayed and the buttons available.

The behavior of the DRE is modeled by ASTRAL transitions. Twelve transitions are specified to model the possible operations of the DRE machine. For instance, the `Insert_PEB` exported transition models the insertion of a qualified PEB device in order to allow various operations to run the election, and the `Initialize_Ballot` transition models the initialization of a ballot when a qualified voter comes.

In a touch-screen based voting system, a voter makes a choice or changes a previous choice by touching the candidate name on the display. In either case, the DRE must capture and process the touch correctly. `Make_Selection` is an exported transition, which must be called by the voter.

```
TRANSITION Make_Selection ( cName: Name )
 ENTRY           [ TIME : M_S_Dur ]
  Which_Phase = during_voting
  & Terminal_Mode = voter_mode
  & Race_Screen ( scrNumber )
  & currentRace = Which_Race(scrNumber)
  & EXISTS C: Candidate ( C ISIN
    Displayed_Candidates (currentRace)
    & Candidate_Name ( C )  = cName )
    & Display( scrNumber ) = Display_Contest(
           Race_Title ( currentRace )
          ,Displayed_Candidates(currentRace)
          ,Screen_Buttons ( scrNumber ) )
   & ~Signal_Enabled & Which_signal = NoSignal
```

specifies the occurrence of a screen touch on a particular candidate's name. On entry, the DRE checks that the voter is voting during voting period, the terminal is in voter mode, the current screen is a race screen displaying both the current race with its candidates and the button(s) required to navigate through the screen, the touched candidate `cName` belongs to the displayed candidates, and that the DRE is not currently sending a signal to the RTAL. We used `Picked` variable to determine whether the candidate has been previously selected. This variable will eventually be used to update the `totalTallyCount` for the selected candidate name `cName` when the ballot is confirmed. The exit assertion for the `Make_Selection` transition is

```
EXIT
 IF ~Picked'( cName,Race_Title(currentRace'))
 THEN
 IF Number_Of_Selected' ( currentRace' ) + 1
 <= Max_Choice_Per_Race ( currentRace )
 THEN /*over-vote is not attempted.*/
  Number_Of_Selected ( currentRace' )  BECOMES
  Number_Of_Selected' ( currentRace' )  + 1
  & Picked ( cName, Race_Title(currentRace') )
  BECOMES TRUE
  & Display ( scrNumber' )  BECOMES
  Update (Display' (scrNumber'),cName,Marked )
  & tempVoteRecord ( currentRace' )  BECOMES
  tempVoteRecord' ( currentRace' )  UNION
   { SETDEF C: Candidate (
       Candidate_Name ( C )  = cName )  }
/*set variable value for the RTAL to print.*/
 & pickedName = cName & pickedValue = Selected
 & Signal_Enabled & Which_Signal = Vote_Signal
 & currentRace = currentRace'
 ELSE /*else over-vote is attempted.*/
  Min_Display ( scrNumber' )  BECOMES
  Display_Info (OverVote_Prohibited, NoButton)
  FI
 ELSE
 /*else, cancel the previous choice.*/
  Number_Of_Selected ( currentRace' )  BECOMES
  Number_Of_Selected' ( currentRace' )  - 1
  & Picked (cName,Race_Title( currentRace' ) )
  BECOMES FALSE
  & Display ( scrNumber' )  BECOMES
  Update (Display'(scrNumber'),cName,UnMarked)
  & tempVoteRecord ( currentRace' )  BECOMES
  tempVoteRecord' ( currentRace' ) SET_DIFF
  { SETDEF C: Candidate (
       Candidate_Name ( C )  = cName )  }
 [...]
```

There are two possible cases when a voter marks a candidate on the screen:

- *Making a selection*. The following scenario occurs: i) as long as there is no overvote attempted the number of selections for this candidate for the current race is incremented by one, `Picked`

is set to true, the current screen is updated, and cName is included in tempVoteRecord, which will be used to display the voter's final selection when the voter requests a preview. In addition, the exported variables pickedName, currentRace, pickedValue and Which_Signal receive new values, and the signaling variable is set to true. This indicates that the RTAL can now print the selection expressed in these exported variables. ii) Otherwise, the voter attempted to overvote and the DRE will display the appropriate message on the screen.

- *Canceling a previous selection*. In this case, the exit assertion specifies that the number of selected candidates for the current race is decremented by one, Picked is reset to false, and cName is removed from the tempVoteRecord. The rest of the variables are updated accordingly and cancellation expressed in these exported variables information are sent to the RTAL.

2) Modeling the Other Processes: The RTAL, PEB and CF Card are respectively specified by the instances of types RTAL_Process, PEB_Process and CFCard_Process.

The RTAL prints vote actions exported by the DRE on a paper tape. The tape contains a list of voter records, where each voter record is a sequence of voter actions. The Tape variable represents the RTAL paper tape where the start information, vote selection, and summary information are continuously printed for each voter. After each print, the RTAL tapePosition is incremented appropriately. The variables RTAL_State and summaryPrinted, respectively, are used for keeping track of the current state of the RTAL and determining whether the summary information has been printed.

Two transitions model the behavior of the RTAL printer. Namely, the Print_Selection transition models behavior of the printing activity, whereas the behavior of the printer after printing the summary information is performing by the transition Scroll_Forward.

As mentioned earlier, the PEB device is used to transfer election specific data between Election Central and poll locations, which is indicated by the variables Candidates_Of_Race tabulatedData, and copyOfBallotImages. While all PEBs are internally identical in construction, they are discernible from one another by the read only information burned in the PIC: their serial number, and more importantly by their PEB kind, namely either "master" or "supervisor".

The most important aspect to specify about the PEB process is that after the terminal is closed, the poll worker uses the master PEB to collect and store the tabulated data and copies of the "images" of the ballots. This is specified by the transition Download_Results.

When the polls are closed, an audit file is saved automatically to the CF Card (by using the Download_AuditData transition). From a formal specification point of view, however, we are only interested in the audit log file, which contains the undervoted races and the number of fleeing voters. These concerns and other behaviors of the CF Card are captured and specified (see the detail in [13]).

B. Critical Security Requirements

Once we specify the relevant information of the system model we need to specify what security requirements the system should meet given the assumptions about the behavior of the system and the external environment that interacts with the system.

In particular, we specified the following concerns:

1) *Environmental/Procedural Assumptions.* We have specified a number of behaviors about the external environment that the e-voting system relies on. For instance, the behavior of the people (voters, pollworkers, and election officials) who interact with the system. This is outside the ES&S system, but it influences how the system operates.
2) *Security Requirements.* We have specified 25 critical requirements (expressed as invariants, constraints, and schedules) that must be satisfied by the system given all the possible assumptions about the environment. In the ES&S system, for instance, the DRE should correctly handle vote selection and the RTAL should update the paper tape after the voter pushes the start button, makes a selection, confirms a vote, or when a poll worker rejects the ballot of a fleeing voter.

We further refine the requirement *G-Req* (see Section III) into the following election result integrity requirements:

- *G-Req1:* The vote entries printed on the RTAL tape during the election must be equal to the cast ballot records plus the rejected vote in the DRE;
- *G-Req2:* After the voting is closed, the results downloaded into the master PEB must be equal to the sum of the results collected from each DRE; furthermore, it must be equal to the sum of the printed paper tapes from all RTALs;
- *G-Req3:* The number of fleeing voters recorded in the audit log file, which is downloaded into the CF card, must be equal to the number of rejected ballots printed on the RTAL tape;
- *G-Req4:* The undervoted races in the audit log file, which is downloaded into the CF card, must be equal to the undervoted races that have been reported on the RTAL tape.

Due to space limitations, we can include only one invariant. *G-Req2* is expressed by the following global invariant:

```
EXISTS p: PEB_Number
  ( the_PEB [p].Kind = Master
  & FORALL d: DRE_Number
```

```
  ( the_PEB[p].ResultDownload_Completed
           ( the_DRE [d].Self )
  & the_DRE [d].Which_Phase = Post_Voting
  & the_DRE [d].DRE_State = Closed
  & FORALL C: Candidate, R: Race
    (C ISIN
      the_DRE [d] .Race_Candidates ( R )
     ->  the_PEB [p].tabulatedData
       (C,R, the_DRE [d].Self )  =
     the_DRE [d].TotalTallyCount(C,R)))))
&
EXISTS p: PEB_Number
 (the_PEB [ p ] .Kind = Master
 & FORALL d: DRE_Number, rt: RTAL_Number
  (the_RTAL [rt]  =
       Plugged_In_RTAL ( the_DRE [ d ]  )
   & the_PEB [p].ResultDownload_Completed
                 (the_DRE[d].Self )
  & the_DRE [d].Which_Phase = Post_Voting
  & the_DRE [d].DRE_State = Closed
  & FORALL C: Candidate, R: Race
   (C ISIN
     the_DRE [ d ] .Race_Candidates ( R )
     ->  the_PEB[p].tabulatedData
      CountCanceled(C,R,the_RTAL[rt])))
```

the first conjunct of the invariant says that there exists a PEB `p`, such that for every DRE `d` in the precinct, if `p` is the master PEB used in `d`'s terminal to download the election results after the `d` terminal is closed and the election has ended (i.e., `Post_Voting` phase), then the election results for each candidate `C` who ran for race `R` stored in `p` is exactly equal to the total tally counted on `d`'s terminal for candidate `C`. Similarly, the second conjunct specifies that for every RTAL printer `rt` and DRE `d` in the precinct, if `rt` is the printer used by `d` during the voting period, then the election result for each candidate `C` who ran for race `R` stored in `p` is the difference between the total number of selected and the total number of canceled votes printed on `rt`.

The rest of the requirements for each process and global specification are specified similarly (refer [13] for more critical requirements).

C. PVS Verification

The specification was first constructed and type-checked using the ASTRAL SDE [14]. We validated the specification and generated the corresponding proof-obligations for the critical requirements. Moreover, the specification was automatically translated into its PVS [9] counterpart using the ASTRAL SDE, which enabled the specification to be passed to the PVS theorem-prover for verification.

Before invoking the theorem prover, the ASTRAL split engine was used to split and classify the ASTRAL specification into collections of simpler properties that infer the whole clause so that the proof of each property could be tackled separately. Table I shows the number of invariants, schedules, and constraints for each of the four processes and the global invariants. It also shows the number after they are split by the ASTRAL SDE.

Table I
NUMBER OF PROOF OBLIGATIONS.

	Proof Obligations	After Splitting
	Invar, Constr, Sched	Invar, Constr, Sched
DRE	4, 6,1	10, 9, 2
RTAL	1, 1, 3	1, 1, 3
PEB	1, 0, 1	2, 0, 1
CFCard	0, 0, 1	0, 0, 2
Global	6, 0, NA	9, 0, NA
Total	12, 7, 6	22, 10, 8

Using the PVS interactive theorem-prover and the techniques discussed in [15] we have proved 13 of the 22 invariants, 3 of the 8 schedules, and 7 of the 10 constraints. We expect that the other global and local properties can be proved using the same or similar proof techniques and strategies.

V. RELATED WORK

Scientific literature on e-voting is wide and multi-disciplinary. Sticking to the topic of this paper, we organize previous work in three different areas: understanding the risks posed by the introduction of e-voting systems in the polling stations; assessing existing systems; designing better e-voting systems using formal techniques.

With respect to the first area, work in the past has focused on understanding what changes could be introduced in the "traditional" voting procedures to allow a secure transition to electronic elections. For instance, [16], [17] discuss risks and difficulties related to the introduction of e-voting, [18], [19] suggest possible improvements to existing procedures, and [20], [21], [22] introduce techniques to formally analyze what security breaches may be derived by executing the procedures in the wrong way. Our work complements those presented above, by helping, e.g., understand how to allocate requirements between procedures and systems to provide more secure electronic elections.

With respect to the second area, assessing existing systems some e-voting systems currently deployed in elections have recently undergone a thorough and independent scrutiny to evaluate their quality. See, for instance, [23], [24], [3], [5], where the authors highlight some serious design and implementation flaws, which could be exploited to compromise elections. The papers also suggest a drastic change in the way in which electronic voting systems are designed, developed, and tested.

With respect to the third area, several works describe the use of formal methods to specify, model, analyze, and assess complex and safety-critical systems (see, for instance, [25], [26], [27], [28] and [29], [30], [31]). The application of formal methods in the e-voting area, however, has been, so

far, limited. We mention [32], [33], that present the formal specification and verification of an electronic voting protocol using π-calculus and [34], [35] where the authors present a tool, FSMC+, that has been used to specify and verify the control logic of an e-voting machine. Our work builds upon and improves those presented above by widening the scope of the analysis and by taking into account aspects related to the procedures and interaction of the system with its environment.

VI. Discussion and Conclusion

In spite of the potential advantages e-voting might bring to the polling station, such as improved turn out, accessibility for impaired people, and improved accuracy and speed, its adoption in various countries has been slow and/or the cause of great debates and controversies.

One of the reasons is that e-voting machines are complex real-time embedded systems required to operate in a (possibly) hostile environment. Another and more relevant reason is the poor design and implementation of (some of) the systems currently deployed for elections in the US and other countries, as different studies have reported and demonstrated. Such weaknesses expose e-voting systems, and consequently elections, to threats and attacks, whose effects vary from a "denial of service" (e.g., stopping the election in a polling station by sabotaging some e-voting machines) to alteration of the results (e.g., by successfully changing votes in some key precincts).

In this paper we have shown how formal verification techniques can be used to model and reason about the security of e-voting systems. The specification was constructed and type-checked using the ASTRAL SDE [14]. The automatically translated specification was passed to the PVS [9] analysis tool. So far we have managed to formally verify that the specification satisfies many of the critical requirements that we discussed in this paper. The proofs were achieved by following the techniques presented in [15]. For instance, we applied the *try-untimed* and *try-untimed-con* proof strategies to prove some of the local invariants and constraints of the system. In general, the proof is carried out first by splitting the critical requirements and applying the appropriate proof strategies developed to support ASTRAL analysis (see [15]).

Although this paper does not consist of a novel specification technique nor a novel voting system, it shows how formal methods can be effectively used for the specification and verification of e-voting systems. In the following paragraphs we discuss two main lessons drawn from this work and some future directions.

The first is that formal methods helps get a better understanding of the security "boundaries" of e-voting systems. In the case of the ES&S, for instance, various security requirements could not be proved without making assumptions about the procedures and about the environment. (Notice that we expect this result to equally apply to other e-voting systems.) Formalizing such hypotheses helps to delineate the necessary conditions for the secure use of systems.

The second is the role open specifications could play for the development of more secure e-voting systems. The formal specification of the ES&S, for instance, required the collection of information from different sources, such as configuration instructions, the user's manual, and videos. The adoption of an open-standardized specification could help simplify, extend, and generalize the results we have found to other systems.

Future work moves along the lines outlined above. Based on the current results, we would like to provide a *generic* specification for DRE-based e-voting machines. This generic specification could then be used as a basis for the specification and design of a new generation of systems (e.g., [36]).

References

[1] T. Kohno, A. Stubblefield, A. D. Rubin, and D. S. Wallach, "Analysis of an Electronic Voting System," *Security and Privacy, IEEE Symposium on*, vol. 0, p. 27, 2004.

[2] J. W. Bryans, B. Littlewood, P. Y. A. Ryan, and L. Strigini, "E-voting: Dependability Requirements and Design for Dependability," in *ARES '06: Proceedings of the First International Conference on Availability, Reliability and Security*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 988–995.

[3] D. Balzarotti, G. Banks, M. Cova, V. Felmetsger, R. Kemmerer, W. Robertson, F. Valeur, and G. Vigna, "Are Your Votes *Really* Counted? Testing the Security of Real-world Electronic Voting Systems," in *Proceedings of the International Symposium on Software Testing and Analysis (ISSTA)*, 2008, pp. 237–248.

[4] Matt Bishop and David Wagner, "Risks of e-voting," *Commun. ACM*, vol. 50, no. 11, pp. 120–120, 2007.

[5] Ryan Gardner and Sujata Garera and Aviel D. Rubin, "On the Difficulty of Validating Voting Machine Software with Software," in *EVT'07: Proceedings of the USENIX/Accurate Electronic Voting Technology*. Berkeley, CA, USA: USENIX Association, 2007, pp. 11–11.

[6] S. of California Secretary of State, "Withdrawal of approval of diebold election systems, inc., gems 1.18.24/accuvote-tswaccuvote-os dre & optical scan voting system and conditional re-approval of use of diebold election systems, inc., gems 1.18.24/accuvote-tsx/accuvote-os dre & optical scan voting system," http://www.sos.ca.gov/elections/voting_systems/ttbr/diebold_102507.pdf, October 2007.

[7] P. McDaniel, M. Blaze, and G. Vigna, "EVEREST: Evaluation and Validation of Election-Related Equipment, Standards and Testing," Ohio Secretary of State's EVEREST Project Report, December 2007.

[8] A. Coen-Porisini, C. Ghezzi, and R. A. Kemmerer, "Specification of Realtime Systems Using ASTRAL," *IEEE Trans. Softw. Eng.*, vol. 23, no. 9, pp. 572–598, 1997.

[9] S. Owre, N. Shankar, and J. M. Rushby, *The PVS Specification Language*, Computer Science Laboratory, SRI International, Menlo Park, CA, February 1993.

[10] E. S. . S. Inc. (ES&S), "iVotronic ™ Voting System," Revision Date: January 2007, version 9.1.x Election Day Operations Checklist.

[11] M. McGaley, "E-voting: an Immature Technology in a Critical Context," Ph.D. dissertation, Dept. of Computer Science, NUI Maynooth, 2008.

[12] R. T. Mercuri and L. J. Camp, "The Code of Elections," *Communication ACM*, vol. 47, no. 10, pp. 52–57, 2004.

[13] K. Weldemariam, R. A. Kemmerer, and A. Villafiorita, "Specification and Analysis of the Electronic Voting Process for the ES&S Voting System," Department of Computer Science, University of California, Santa Barbara, Tech. Rep., March 2009.

[14] P. Z. Kolano, "ASTRAL Software Development Environment User"s Manual," Santa Barbara, CA, USA, Tech. Rep., 1997.

[15] P. Kolano, "Tools and Techniques for the Design and Systematic Analysis of Real-Time Systems," Ph.D. dissertation, University of California, Santa Barbara, 1999.

[16] A. Xenakis and A. Macintosh, "G2G Collaboration to Support the Deployment of e-Voting in the UK: A Discussion Paper," in *EGOV*, ser. Lecture Notes in Computer Science. Springer, 2004, pp. 240–245.

[17] ——, "Levels of difficulty in introducing e-voting," in *EGOV*, ser. Lecture Notes in Computer Science. Springer, 2004, pp. 116–121.

[18] M. Volkamer and R. Krimmer, "Independent audits of remote electronic voting – developing a common criteria protection profile." in *Proceedings der EDEM 2007 – Elektronische Demokratie in Österreich*, 2007.

[19] A. Prosser, R. Kofler, R. Krimmer, and M. K. Unger, "Security assets in e-voting," in *Electronic Voting in Europe*, 2004, pp. 171–180.

[20] A. Xenakis and A. Macintosh, "Procedural Security Analysis of Electronic Voting," in *ICEC '04: Proceedings of the 6th international conference on Electronic Commerce*. New York, NY, USA: ACM Press, 2004, pp. 541–546.

[21] K. Weldemariam, A. Villafiorita, and A. Mattioli, "Assessing Procedural Risks and Threats in e-Voting: Challenges and an Approach," in *VOTE-ID*, ser. Lecture Notes in Computer Science. Springer, 2007, pp. 38–49.

[22] K. Weldemariam and A. Villafiorita, "Modeling and Analysis of Procedural Security in (e)Voting: The Trentino's Approach and Experiences," in *EVT*. Berkeley, CA, USA: USENIX Association, 2008.

[23] N. Ansari, P. Sakarindr, E. Haghani, C. Zhang, A. K. Jain, and Y. Q. Shi, "Evaluating electronic voting systems equipped with voter-verified paper records," *IEEE Security and Privacy*, vol. 6, no. 3, pp. 30–39, 2008.

[24] A. Aviv, P. Cerny, S. Clark, M. S. Eric Cronin, Gaurav Shah, and M. Blaze, "Security Evaluation of ES&S Voting Machines and Election Management System," in *In Proc. of the USENIX/ACCURATE Electronic Voting Technology Workshop*, 2008.

[25] Z. Dang and R. A. Kemmerer, "A Symbolic Model Checker for Testing ASTRAL Real-Time Specifications," *Real-Time Computing Systems and Applications, International Workshop on*, vol. 0, p. 174, 1999.

[26] R. Corin, S. Etalle, P. H. Hartel, and A. Mader, "Timed model checking of security protocols," in *FMSE '04: Proceedings of the 2004 ACM workshop on Formal methods in security engineering*. New York, NY, USA: ACM, 2004, pp. 23–32.

[27] N. Markey and P. Schnoebelen, "Tsmv: A symbolic model checker for quantitative analysis of systems," *Quantitative Evaluation of Systems, International Conference on*, vol. 0, pp. 330–331, 2004.

[28] P. Bertoli, A. Cimatti, M. Pistore, M. Roveri, and P. Traverso, "NuSMV 2: An Open Source Tool for Symbolic Model Checking," in *Proc. of International Conference on Computer-Aided Verification*, 2002.

[29] M. Lowry and D. Dvorak, "Analytic Verification of Flight Software," *IEEE Intelligent Systems*, vol. 13, no. 5, pp. 45–49, 1998.

[30] Z. Dang and R. A. Kemmerer, "Using the ASTRAL model checker to analyze mobile IP," in *ICSE '99: Proceedings of the 21st international conference on Software engineering*. Los Alamitos, CA, USA: IEEE Computer Society Press, 1999, pp. 132–141.

[31] C. Braghin, N. Sharygina, and K. Barone-Adesi, "Automated Verification of Security Policies in Mobile Code," in *IFM*, ser. Lecture Notes in Computer Science. Springer, 2007, pp. 37–53.

[32] S. Delaune, S. Kremer, and M. D. Ryan, "Verifying Privacy-type Properties of Electronic Voting Protocols," Laboratoire Spécification et Vérification, ENS Cachan, France, Research Report LSV-08-01, Jan. 2008, 55 pages.

[33] S. Kremer and M. D. Ryan, "Analysis of an electronic voting protocol in the applied pi-calculus," in *ESOP'05*, vol. 3444. Springer, 2005, pp. 186–200.

[34] R. Tiella, A. Villafiorita, and S. Tomasi, "Fsmc+, a tool for the generation of java code from statecharts," in *PPPJ '07: Proceedings of the 5th international symposium on Principles and practice of programming in Java*. New York, NY, USA: ACM, 2007, pp. 93–102.

[35] Adolfo Villafiorita, Komminist Weldemariam and Roberto Tiella, "Development, Formal Verification and Evaluation of an eVoting System with VVPAT," *IEEE Transaction on Information Forensics and Security*, vol. 4, December 2009.

[36] Cynthia Sturton and Susmit Jha and Sanjit A. Seshia and David Wagner, "On voting machine design for verification and testability," in *ACM Conference on Computer and Communications Security*, Ehab Al-Shaer and Somesh Jha and Angelos D. Keromytis, Ed., vol. 463-476. ACM, 2009.