The design, implementation, and evaluation of an Internet-based eVoting system*

Christos Manolopoulos[1], Anastasia Panagiotaki[1], Dimitris Sofotassios[1], Paul Spirakis[1], Yannis C. Stamatiou[1,2]

[1]Research Academic Computer Technology Institute,
N. Kazantzaki, University of Patras, 26500, Greece. emails: {manolop, panagioa,sofos,spirakis}@cti.gr
[2]University of Ioannina, Mathematics Department,
45110, Ioannina, Greece. email: istamat@uoi.gr

*This work has been partially supported by the General Secretariat of Research and Technology of Greece, under the project PNYKA (project code ΔΕΛ_2, decision 8948/04.05.06), Operational Program of Western Greece, 3rd Community Support Framework

Abstract

In this paper, we describe the stages for the design, implementation and evaluation of a complete, Internet-based eVoting system for performing large-scale elections. Our focus was on the application of trust modeling and risk assessment methodologies in conjunction with strong cryptographic protocols and open source tools for improving the openness of the system to public scrutiny. The resulting system has been successfully evaluated during an in-field trial process that involved a mock-up election held for the Technical Chamber of Greece.

Keywords: eVoting, security protocol, cryptographic protocol, distributed system

1. Introduction

In sharp contrast to a number of Internet-based applications that have attracted peoples' trust over the past few years (e.g. tax declaration, eGovernment-related form completion etc.), the widespread adoption of eVoting technologies still seems to be out of reach. This can be attributed, mainly, to two reasons: a) Incidents of malicious interventions or accidental faults in the system occur that raise suspicions about the robustness and credibility of the system, b) The system designers and implementers produce scant documentation with regard to critical issues and sensitive parts of the system, as opposed to producing usually massive information with regard to system installation, operation, maintenance and fine tuning. ACM's special issue on eVoting (see [CACM (2008)]) gives an excellent and in depth account of the general issues involved in people's mistrust in eVoting systems.

There is much ongoing research in the development and analysis of new trust management models for complex and dependable computer systems. [Blaze *et al*.

(1996)] proposed the application of automated trust mechanisms in distributed systems. In [Josang (1996)] the focus is on the strong relationship between the notions of trust and security. The composition and propagation of trust information between elements of information systems is also of pivotal concern and a number of research works are devoted to them (see [Richardson et. al. (2003), Kamvar et. al. (2003), Guha et. al. (2004)]).

As far as trust in the eGovernment field is concerned, there are efforts targeted at trust models based on distributed trusted agents such as the PKIs (see [Tassabehji, R. and Elliman (2006)]). There are many open issues both conceptual and practical, however, that pertain to eGovernment trust, many of which are discussed in [Kim et. al. (2005)]. There are fewer attempts, however, for trust management in the eVoting field. Due to the complexity of an eVoting system, most efforts are focused on the study of specific system security requirements such as, for instance, establishing uncoercibility of the voters ([Acker (2004)]. Also, as a common practice for strengthening trust, many approaches focus on the existence of a voter verifiable paper copy of the ballot or the design of strong cryptographic protocols (e.g. [Gritzalis (2003), Smith (2006)]. Finally, the work done by the OASIS consortium [OASIS (2006)] is a first step towards the standardization of secure eVoting architectures based on formal modelling and risk assessment methodologies (e.g. use of the EML language and threat evaluation techniques).

In our efforts to implement an eVoting system that handles these two issues (as required by a project funded by the Greek Secretariat of Research Technology), we applied a design and implementation methodology aiming at establishing trust among people towards the final system (see [Antoniou et. al. (2007)]. Thus, the system was designed and implemented guided by formal risk analysis and managements processes, used strong cryptographic protocols, and employed only open source development tools. Moreover, simulations were conducted as well as a trial using a mock-up election process in order to provide evidence of its efficiency and security. In what follows we detail the stages of the adopted methodology showing samples of the trust-evidence they produce along the way.

2. System architecture and elements

The main architectural features of the target system are the following: (i) A highly distributed architecture for efficiency and control sharing: an hierarchy of central and local Election Authorities with distributed computations within an Election Authority, as depicted in Figure 1. (ii) A robust voting protocol that ensures the basic voting security requirements (secrecy, receipt-freeness, uncoercibility, verifiability, etc.). The protocol is based on strong cryptographic primitives, including zero-knowledge proofs that, essentially, provide the guarantees (without violating the vote secrecy requirement) that votes are correctly received and included in the voting outcome.

The adopted eVoting protocol is the protocol described by Warren Smith in [Smith (2006)] which is based on the homomorphic properties of the El Gamal encryption function and the hardness of computing the discrete logarithm (see [Lenstra and Lenstra (1990)] for complexity theoretic issues related to this problem).

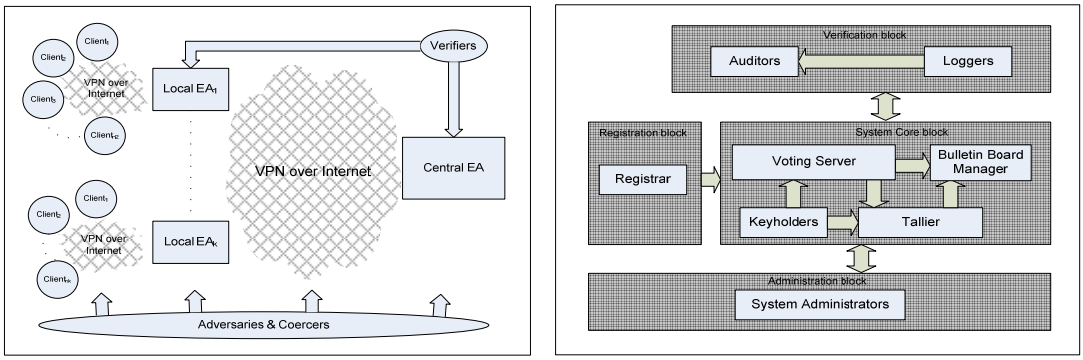

Figure 1. *(a) eVoting system architecture (b) Modules of an Election Authority (EA)*

3. Design and implementation

3.1 Trust architecture and risk analysis

Our approach relies on two general methodologies and one eVoting specific protocol. The two methodologies is the *layers of trust decomposition* of a system (see [KLSSY04, [KLSSY05]) and the *CORAS risk assessment framework* for security critical systems (see [CORAS03]). Below we will provide a brief account of these two elements of the approach

The layers of trust view of the eVoting system is a view complementary to the other formal views and models of ordinary IT systems (e.g. business view, technical view etc.) and is employed in order to handle the complexity of the security issues pertaining to eVoting, as defined by its security requirements. This complexity can be as high as the complexities that arise in other architectural views of such systems and the layers of trust approach can be used as a tool for managing these issues successfully.

The role of the layers, and the correspondence to the e-voting system, is as follows (see [Konstantinou et. al. (2005)]: 1) *Scientific soundness*: All the components of the system should possess some type of security justification (strong cryptographic primitives) and be widely accepted within the scientific community. 2) *Implementation soundness*: A methodology should be adopted that will lead to the verification of the implementation of the separate system components as well as the system as a whole. In addition, such a verification methodology should be applied

periodically to the system. 3) *Internal operation soundness*: The design and implementation should offer high availability and fault tolerance and should support system self-auditing, self-checking, and self-recovery from malfunction. 4) *Externally visible operational soundness*: It should be possible for everyone to check log and audit information at some level. 5) *Convincing the public (social side of security)*: It is crucial for the wide acceptance of the eVoting system that the public will trust it when it is in operation. This trust can be, in general, amplified if the eVoting authority publicizes the details of the design and operation of the eVoting system to the public.

With regard to risk analysis, it is complementary to the trust architecture outlined above and it is applied to identify and treat risks that may exist in the target system. We applied the CORAS risk analysis methodology (see [Stølen et. al. (2003)] which is based on the RM-ODP standard ([Putman (2000)]). The first step is the *Context Identification*. This step involves a detailed description of the system under study (application scenario, assets, data flows) using the UML modeling language (see, e.g., [Krutchten (1999)]). The description uses various different types of UML diagrams, depicting different system aspects. The aim is to gain a good understanding of the system and document it using visual means. The diagrams that we used include *Use case diagrams* that show system functionality, *Activity Diagrams* that describe workflows, *Time Sequence Diagrams* (see example in Figure 2) that describe the exchange of data among stakeholders per time. System assets were also evaluated with regard to their criticality.

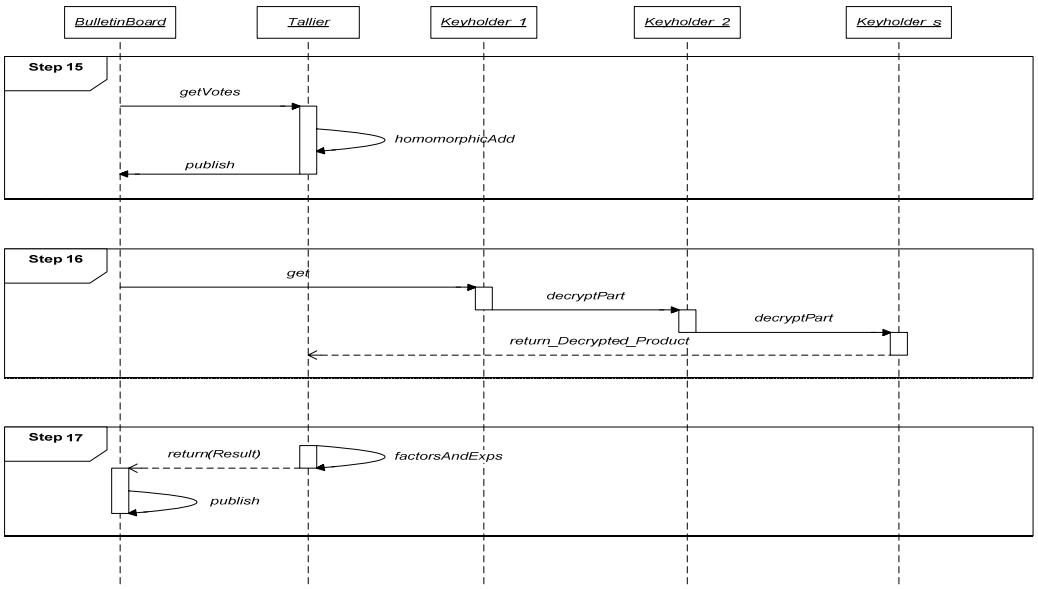

Figure 2. *Time Sequence Diagram*

The next step is the *Risk Identification*. This step aims at the identification and documentation of the threats that the system faces, using appropriate *Threat Diagrams*. A HazOp analysis is performed to provide a first level assessment of threats and propose initial countermeasures. For the most critical threats among the ones identified, a Fault Tree Analysis is performed to identify events that cause the specific threats. Two examples of the diagrams used in this step are shown in Figures 3 and 4.

Threat (accidental) / Threat (deliberate) / Threat (non-human)	Threat Scenario / Unwanted Incident / Asset	Vulnerability
Who/what causes it?	**How? What is the incident? What does it harm?**	**What makes it possible?**
Keyholders	Disclosure of secret keys	Corrupted Keyholders (software)
Voter	Disclosure of credentials (id, password, πιστοποιητικό) to another person	Malicious Voter
EA	Vote Alteration	Corrupted EA
EA	Vote disclosure	Corrupted EA
EA	Tallying error	Software Error
EA	Result Alteration	Corrupted EA
Coercer	Voter coercing	Lack of monitoring during remote vote casting
Hacker	Vote Alteration	Insufficient Security
Hacker	Final result Alteration	Insufficient Security

Figure 3. *Part of High-level Risk Table*

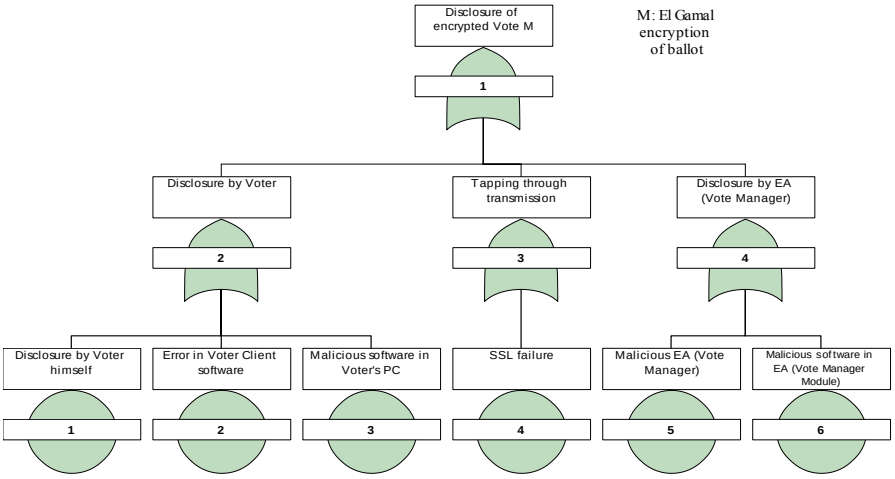

Figure 4. *Fault Tree Diagram*

Then we apply the *Risk Analysis* step. This step aims at estimating the risks that are caused by the threats identified in the previous step. At first, we defined the levels of scale of the various sizes that are used in the risk analysis (i.e. likelihood of event

occurrence, consequence and risk). Then, we estimated the amount of risk (quantitatively or qualitatively using Fault Tree Analysis). Tables 1 and 2 presented below are parts of the full tables compiled during our analysis.

Table 1. *Assessment of likelihood of occurrence of unwanted incidents*

Event	Description	Likelihood
Disclosure by Voter		
1	Disclosure of Vote by Voter	0,05
2	Voter software error	0,1
3	Malicious software in Voter's PC	0,1
Tapping during transmission		
4	SSL failure	0,1
Disclosure by Vote Manager		
5	Malicious Election Authority (vote manager)	0,05
6	Malicious software in Election Authority (vote manager)	0,05

Table 2. *Qualitative assessment of Consequence using FMEA*

ID	Function/ Entity	Failure Mode	Effects		Causes	Consequences
			Local	System wide		
1	GenerateElGamalParameters (size)	*Size* parameter is not available in system *config* file	The public parameters may not be created	System initialization is not possible	Config file is not properly updated by system administrator. Access to config file/database is not possible	Voting process may not begin
2	Publish(elGamalParameters)	*Bulletin Board* is not updated with the public parameters	Keyholders may not produce keys	System initialization is not possible	Connection to database is not possible	Voting process may not begin

Then we have the *Risk Evaluation* step. This step includes the evaluation of system risk, based on the previous analysis and the risk levels defined. The evaluation results are presented in the risk categorization matrix, as shown in the Table 3.

Table 3. *Risk Categorization Matrix.*

Consequence Value	Likelihood Value				
	Rare	Unlikely	Possible	Likely	Certain
Insignificant					
Minor		4, 10, 12, 30, 31	29, 32, 34, 35, 36, 39, 40	14	
Moderate		3	8, 22		
Major		1, 9, 21, 23, 26, 27	7, 17 , 20, 24, 25, 28, 33, 37	13	
Catastrophic	2, 5, 11, 47	6, 15, 16, 18, 19, 41, 43, 44, 45, 46	38, 48, 49	42	

The numbers appearing in Table 3 correspond to the risks that were identified and analyzed in the previous steps. The risks are classified from acceptable (white area) to

extreme (dark gray area). For instance, risk No 42 corresponds to "Multiple, timestamped votes do not arrive in the same sequence they were submitted". This risk is considered to be extreme, since it has high occurrence likelihood and catastrophic consequences.

Finally, we have the *Risk Treatment* step. This last step of the methodology involves decisions to be taken about prioritizing and treating the identified risks, based on the aforementioned categorization. Specific countermeasures are proposed for each risk, with emphasis on extreme risks. Table 4 is an excerpt of the risk treatment table, showing some security risks and the proposed treatments.

Table 4. Risk Treatment Table

Risk ID	Description	Risk Level	Treatment options - measures
Risks with regard to Partial Keys disclosure or non-availability			
2	Disclosure of some of K_i by their keyholders	Extreme	The disclosure of partial keys would be catastrophic, as it would allow the decryption of individual votes and the final result by unauthorized parties (or even the EA)
5	Some of the K_i are not available	Extreme	**Threshold cryptography techniques** are used as a countermeasure. Such techniques require for at least t out of n keyholders to cooperate for the conduction of the elections. Moreover, colluding interests of the keyholders discourage potential alliances among them. For ultimate security, we suggest that $t=n$, which means that all keyholders need to cooperate.
Risks with regard to votes submission order			

Some of the conclusions reached through the application of CORAS were the following. In general, the adopted voting protocol, as well as many of the engineering decisions made during the design of the system, was proved to be reasonable choices since it was found to handle well all the threats discovered through the application of CORAS. The protocol employs threshold cryptography, suitably time-stamped multiple votes, use of Zero Knowledge Proofs for validating the encrypted votes etc. Also, the analysis indicated aspects of the initial design that were further enhanced (e.g. web access vs. client server access, SSL vs. VPN between the voter and the Election Authority, etc.).

3.2 System components

The full system is comprised of a number of cryptographic libraries, open source tools related to IT security and a number of web interfaces that allow system set-up and initialization as well as vote submission.

The implementation of the central EA is based on the following: Ubuntu Linux, Java Runtime Environment (JRE), PostgreSQL Server, Java crypto libraries (Bouncy Castle), JDBC Driver for PostgreSQL, Server clock synchronization with a legal NTP server, OpenVPN (VPN Server), and OpenCA. The implementation of the local EAs

are based on the following: Ubuntu Linux, Apache Tomcat 5.5, Java crypto libraries (based on Bouncy Castle), JDBC Driver for PostgreSQL, Server clock synchronization with an authorized NTP (Network Time Protocol) server, and OpenVPN (VPN Client). With regard to the web interfaces, they are based on three-tier architecture: *Presentation*, *Application* and *Data Tier*. The presentation tier is related to the web browser, the application tier is related to the application server (Apache Tomcat in our case) and the data tier is related to the database server (PostgreSQL in our case). The web interface implementation is based on the technology of Java Server Pages (JSP). In order to set-up and operate a voting process, the following components must be activated: VPN connections between the central and the local EAs, the OpenCA software on the central EA, and the Apache Tomcat software item on the local EAs. First the voter sends a request to the Tomcat application server which is appropriate local EA. The server executes the code and sends the responses to the voter. The communication is realized using the HTTPS protocol. The certificate for this communication is distributed by the EA's Certification authority. The voter submits his vote and the vote is encrypted and stored in the local EA's database. Simultaneously, the vote is forwarded to the central EA's database. The communication between local and central EAs' databases is realized through a VPN.

4. System evaluation

4.1 Performance simulation

In order to evaluate the dynamics as well as performance of the distributed eVoting architecture, we modeled it as an *open Jackson network of queues*. This type of networks is characterized by constant expected incoming packet rates and exponentially distributed service rates. The constant incoming packer rate, however, is not realistic for networks supporting elections, If we assume, for instance, that the voting process spans an interval from 8:00 to 20:00, then we expect a low voter arrival rate in the morning that reaches a peak around noon and then decreases up to the time when the election ends. That is, the expected arrival rate is not uniform over the voting period but, rather, follows a *unimodal distribution*. This behaviour is supported from empirical evidence although a statistical analysis during a real election process should be performed for accurate results. For the purposes of our project we have developed a simulation tool in C based on the simulation package CSIM 19 of Mesquite. This tool can be used for modelling and simulating any distributed architecture based on the open Jackson model. The tool offers a wide range of user-settable model parameters for describing the characteristics of the distributed architecture as well as performance metrics.

4.2 Trials

The objective of the pilot was to assess the overall operation of the system, to validate the adopted architecture and to test the integration among the chosen third-party open source tools. A further objective was to obtain a feedback from actual users/voters with regard to system functionality and security. Finally, the pilot was used as a vehicle towards a gradual introduction of the system to election scenarios of a larger scale. With regard to the mock-up election process, the goal was engage 200 of the members of the Western sector of the Technical Chamber of Greece. This group was selected due to the easiness of conducting its members and the fact that, being engineers themselves, they could follow the voting instructions easily and pinpoint technical difficulties with the system. In addition, the Technical Chamber is interested in automating its voting procedures and, thus, it was a good target for the system trials. The overall impression was positive and the feedback received is already under consideration for further improvements and enhancements of the system.

5. Discussion

In this paper we summarized the design, implementation, and evaluation methodologies that led to the development of a secure, distributed eVoting platform capable of supporting large scale election processes. Our main efforts were towards the increase of trust-related propertied of the final system. To this end, we used formal design and risk analysis methodologies that, also, produced ample documentation and system models along the way, so that the final system would be easy to inspect and understand. In addition, the whole system was based on open source tools and software. The system has been successfully used to perform a mock-up election process set-up for the Technical Chamber of Greece. We believe that our approach can be further formalized and used in other efforts for building secure eVoting systems with an eye towards public system verifiability and transparency.

References

Antoniou, A., Korakas, C., Manolopoulos, C., Panagiotaki, A., Sofotassios, D., Spirakis, P. and Stamatiou, Y. C., *A Trust-Centered Approach for Building E-Voting Systems*, Lecture Notes in Computer Science, Volume 4656, 2007, pp. 366-377, Springer Berlin / Heidelberg.

Acker, B. van, Remote e-Voting and Coersion: A risk Assessemnt Model and Solutions, in: *Electronic Voting in Europe - Technology, Law, Politics and Society*, LNI Proc., pp. 53 – 62, GI-Editions, 2004.

Blaze, M., Feigenbaum, J., and Lacy, J., Decentralized trust management, in: *Proc. IEEE Symposium on Security and Privacy*, Oakland (CA, USA), 1996, pp. 164-173, 1996.

CACM, The problems and potentials of voting systems, Communications of the ACM, Special Issue on eVoting 47(10), October 2004.

Gritzalis, D.A., Secure Electronic Voting, Series: Advances in Information Security, Vol. 7, Kluwer Academic Publishers, 2003.

Guha, R., Kumar, R., Raghavan, P., and Tomkins, A., Propagation of trust and distrust, in: *Proc. International Conference on WWW*, pp. 403-412, 2004.

Krutchten, P., *The Rational Unified Process, An Introduction*, Addison-Wesley, 1999.

Josang, A., The right type of trust for distributed systems, in *Proc. New Security Paradigms Workshop*, pp. 119-131, 1996.

Kamvar, S.D, Schlosser, M.T., and Garcia-Molina, H., The eigentrust algorithm for reputation management in p2p networks, in: *Proc. International Conference on World Wide Web,* pp. 640-651, 2003.

Lenstra, A.K. and Lenstra., H.W. JR., Algorithms in number theory, in: J. van Leeuwen, ed., Handbook of Theoretical Computer Science, vol. A North-Holland, Amsderdam, pp. 673-715, 1990.

Putman, J. R., *Architecting with RM-ODP*, Prentice-Hall, 2000.

Richardson, M. Agrawal, R. and Domingos, P. Trust management for the semantic web, in: *Proc. International Semantic Web Conference*, pp. 351- 368, 2003

Smith, W. D., *Cryptography meets voting,* living document, version of January 2006.

Kim D.J. Song Y.I. Braynov S.B. Rao H.R., A multidimensional trust formation model in *B-to-C e-commerce: a conceptual framework and content analyses of academia/practicioner perspectives, in: Decision Support Systems*, 40, pp.143-165, 2005.

Konstantinou, E., Liagkou, V., Spirakis, P., Stamatiou, Y., and Yung, M., Trust Engineering: from requirements to system design and maintenance – a working national lottery system experience, in: *Proc. Information Security Conference - ISC 2005*, LNCS 3650, Springer Verlag, pp.44-58, 2005.

OASIS Standard, EML Process and Data Requirements, ver 4.0, February 2006.

Stølen, K., den Braber, F., Dimitrakos, T., Fredriksen, R., Gran, B.A., Houmb, S.-H., Stamatiou, Aagedal, J. Ø, Model-based risk assessment in a component-based software engineering process: the CORAS approach to identify security risks, Chapter in Business Component-Based Software Engineering, Franck Barbier (ed), Kluwer, 2003, pp. 189-207.

Tassabehji, R., and Elliman, T., Generating citizen trust in e-government using a trust verification agent: a research note, in: *CD-ROM/Online Proceedings of the European and Mediterranean Conference on Information Systems (EMCIS)*, Costa Blanca, Alicante, Spain, 2006.