EXPERIENCES AND BENEFITS FROM THE APPLICATION OF A FORMAL RISK ASSESSMENT FRAMEWORK IN THE EVOTING DOMAIN

Christos Manolopoulos[1], *Anastasia Panagiotaki*[2], *Dimitris Sofotassios*[3], *Paul Spirakis*[4], *Yannis Stamatiou*[5]

[1, 2, 3, 4] Research Academic Computer Technology Institute, N. Kazantzaki, University of Patras, 26500, Rio, Patras, Greece, manolop@cti.gr, panagioa@cti.gr, sofos@cti.gr, spirakis@cti.gr

[4] Computer Engineering and Informatics Department, 265 00, Patras, Greece

[5] Mathematics Department, 451 10, Ioannina, Greece, istamat@cc.uoi.gr

Abstract: *A critical success factor for an information system in the eGovernment domain is to attract and maintain people's trust in its capabilities and operation correctness. This factor is especially important in the eVoting domain, which is one of the most sensitive eGovernment components. In this paper we present our experience from the application of a formal risk analysis framework for ICT based systems, the CORAS framework, in order to support the development of an Internet-based eVoting system. CORAS permeates the development process in all the layers of the target system and aims at the precise, unambiguous, and efficient risk assessment during their design, implementation and operation phases. It focuses on the integration of viewpoint-oriented UML-like modeling and employs a variety of risk analysis methods. In addition, CORAS produces a detailed system documentation which can be publicized in order to increase transparency, providing thus an open view of the system to the public leading to its wider acceptance, in contrast with most "closed-design" commercial eVoting systems.*[*]

1. Introduction

The rapid increase of penetration of the Internet and information technology into people's lives in conjunction with the desire for more free time available for non-professional activities

[*] This work has been partially supported by General Secretariat of Research and Technology of Greece, under the projects:

- ΑΚΜΩΝ (project code 01 ΑΚΜΩΝ 23, decision 296/ΕΦΑ 31/11.01.2005), Operational Program "Competitiveness" 2000-2006, 3rd Community Support Framework.
- ΠΝΥΚΑ (project code ΔΕΛ_2, decision 8948/04.05.06), Operational Program of Western Greece, 3rd Community Support Framework.

has caused a need for providing citizens with the ability to access government services and participate in political decision making *remotely*, especially through the widely available and used Internet infrastructure. With regard to remote participation, *Electronic Voting*, or eVoting, has been attracting lately the attention of many states and researchers as the vehicle for increasing citizens' participation and strengthening the democratic processes, realizing the concept of *eParticipation*.

One obstacle, however, that prevents the widespread use of eVoting technologies is the lack of people's trust towards the systems that implement these technologies. 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 [1] gives an excellent and in depth account of the general issues involved in people's mistrust in eVoting systems.

In this paper we present our experience from the application of the CORAS *formal risk analysis framework* that handles efficiently and effectively these two issues, leading to an eVoting system design that is well documented with regard to security critical aspects as well as free of as many as possible vulnerabilities even from the early design and implementation phases. The effect of this is that citizens would have a tangible way of accepting an eVoting system and, thus, increase their participation in electronic voting procedures. In addition, external system auditors would have at their disposal a clear description of the eVoting system, its components and the security critical issues in order to conduct their system audit/certification activities.

2. Brief description of the CORAS framework

CORAS is a framework for model-based risk assessment targeting security critical systems that was the outcome of an EU funded RTD project (IST-2000-25031). The CORAS platform (available at http://coras.sourceforge.net/, see [13] for a more thorough description) has four main anchor-points: (1) a risk documentation framework based on the Reference Model for Open Distributed Processing [11] (RM-ODP), (2) a risk management process based on AS/NZS 4360 [3], (3) an integrated risk management and system development process based on the Unified Process (UP) [9] and (4) a platform for tool-integration based on XML [8]. The risk documentation framework is used both to document the target of assessment as well as to record risk assessment results. The risk management process provides a sequencing of the risk management process into sub-processes for context establishing, risk identification, risk assessment, risk evaluation, and risk treatment.

One of the central ideas behind the CORAS methodology is the use of semi-formal graphical description techniques. The descriptions are mainly expressed in the Universal Modelling Language (UML). Moreover, CORAS is a careful integration of techniques and templates inspired by Hazard and Operability (HazOp) Analysis [9], Fault Tree Analysis (FTA) [6], Failure Mode and Effect Criticality Analysis (FMECA) [7], Markov Analysis ([2],[12]) as well as CRAMM [5].

3. Brief description of the target system

The selected target platform is an eVoting system that was developed within a nationally funded research project. It is an Internet-based system and supports a wide range of voting processes, from polling procedures to large scale election processes and referenda. Its main features include:

- 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 the Figure below:

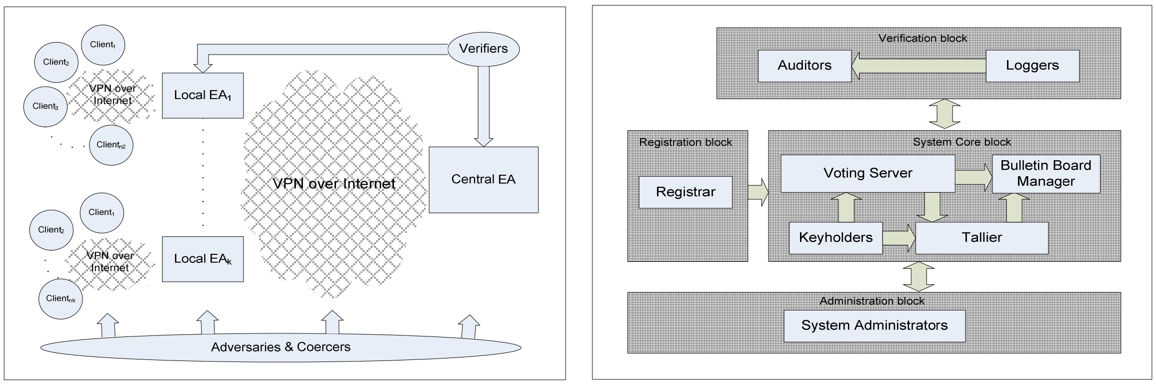


Figure 1: (a) eVoting system architecture (b) The modules of an Election Authority (EA)

- 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 protocol uses Elgamal homomorphic encryption and it is based on multiparty computations and threshold cryptography, involving mutually distrusting agents, called keyholders, who control the voting process. The interested reader may consult [13] for the technical details and proofs of security of the protocol.

The application of the risk assessment framework to the eVoting platform does not aim to *prove* the scientific soundness of the protocol used (which is already theoretically proven) but rather to control the soundness of its actual implementation and the integration to the overall system design. Our approach was part of an overall "trust engineering" methodology followed for building the system as described in [3].

4. The application of CORAS in the eVoting system

The risk analysis addresses the security aspects of the eVoting platform, in order to:

- Identify the security risks associated with the usage of the system for Internet-based voting.
- Advise on whether the usage of the system is in accordance with applicable legislation, standards, recommendations and guidelines.
- Recommend the additional security related improvements that have to be provided.

The focus of the risk analysis was on the voting protocol and its implementation since this is one of the most crucial components of the system. More specifically, our analysis targets were the following:

- *Data:* The transmission of data between the voters and the distributed servers.
- *Procedures:* The protocol steps executed between the client and the server modules.
- The threats associated with corrupted keyholders (i.e. the holders of the shares of the voting key).
- The risks related to vote disclosure.

We did not handle the following issues, since they are not directly involved in the voting protocol itself but, rather, on the infrastructure in which the protocol operates:

- The risks associated with the interfaces of the platform with other internal applications.
- The internal handling and processing of the stored data by other applications/processes running on the same computer as the eVoting system modules.
- The risks associated with the network components (e.g. malfunctioning routers).
- The physical threats to the system infrastructure (e.g. broken communication lines or damage on the servers due to flooding in the buildings).

The steps that were carried out, as prescribed by CORAS, are presented in the following sections.

4.1 Context Identification

This step involves a detailed description of the system under study (application scenario, assets, data flows) using the UML modeling language. 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.

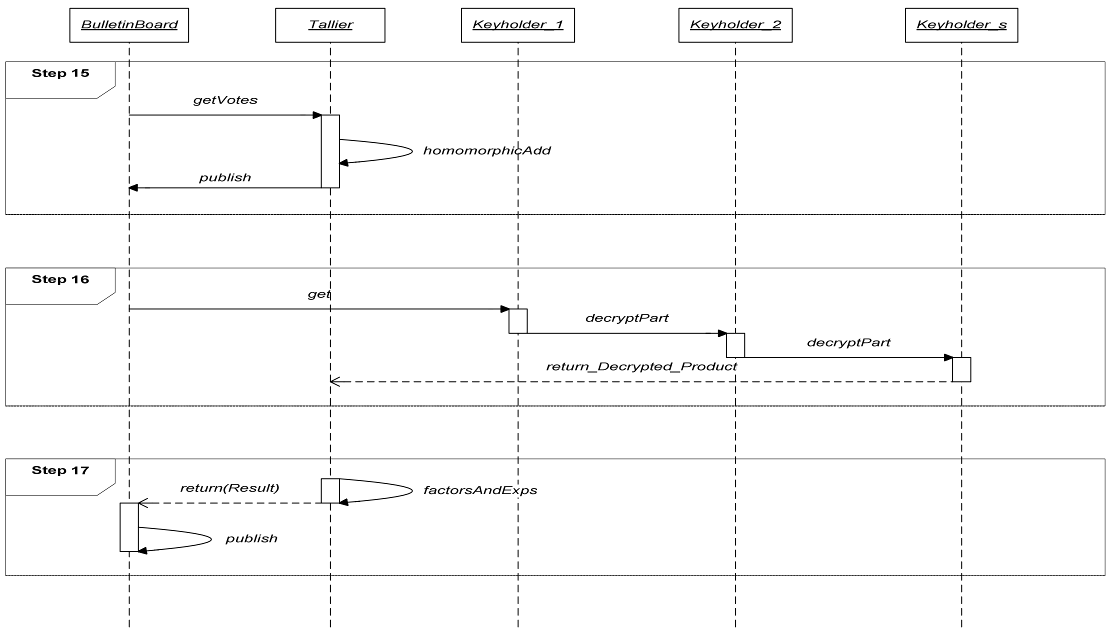


Figure 2: Time Sequence Diagram

4.2 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 (using the ITEM Toolkit software, http://www.itemuk.com/) to identify events that cause the specific threats. Two examples of the diagrams used in this step are shown below:

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

Table 1: Part of High-level Risk Table

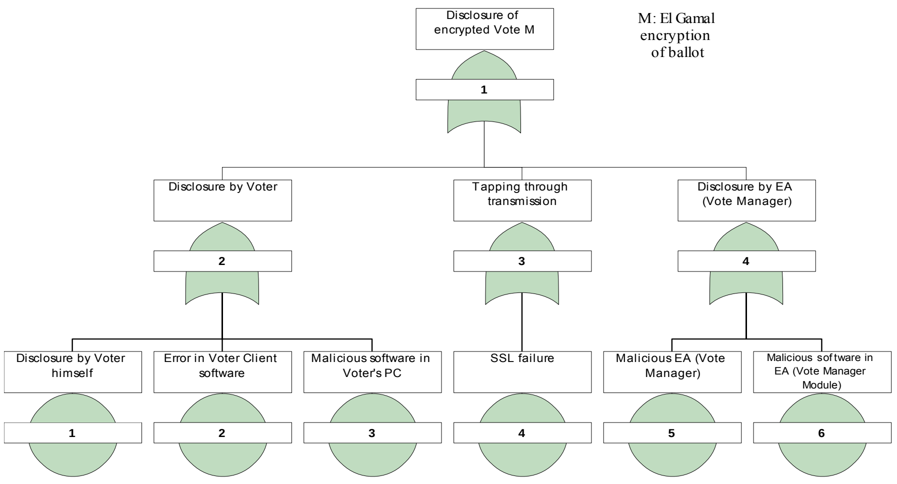

Disclosure of encrypted Vote M
M: El Gamal encryption of ballot
1
Disclosure by Voter
2
Tapping through transmission
3
Disclosure by EA (Vote Manager)
4
Disclosure by Voter himself
1
Error in Voter Client software
2
Malicious software in Voter's PC
3
SSL failure
4
Malicious EA (Vote Manager)
5
Malicious software in EA (Vote Manager Module)
6

Figure 3: Fault Tree Diagram (ITEM Toolkit)

4.3 Risk Analysis

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). The tables presented below are parts of the full tables compiled during our analysis.

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: Assessment of likelihood of occurrence of unwanted incidents

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

Table 3: Qualitative assessment of Consequence using FMEA (ITEM Toolkit)

4.4 Risk Evaluation

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 below.

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**	

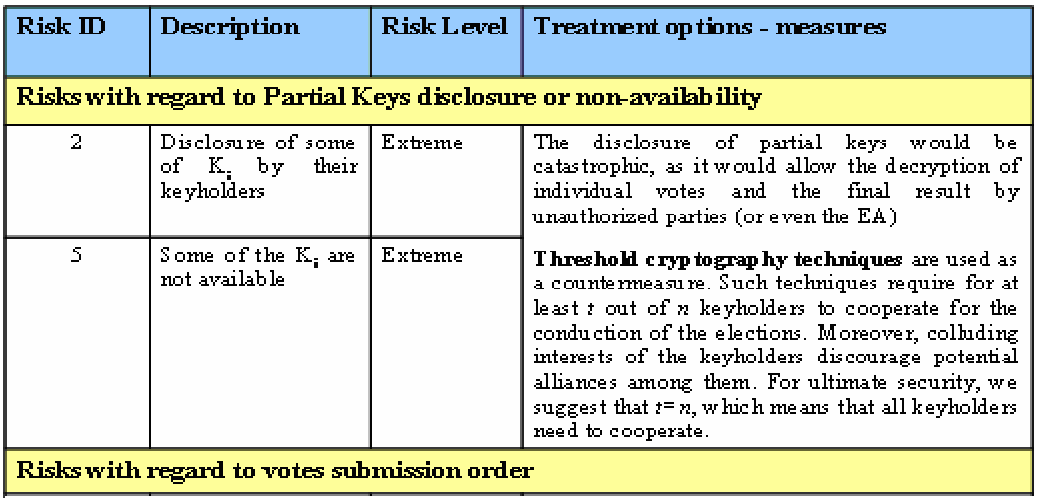
Table 4: Risk Categorization Matrix

The numbers appearing on Table 4: correspond to the risks (risk IDs) 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 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.

4.5 Risk Treatment

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 5: below is an excerpt of the whole risk treatment table, showing some security risks and the proposed treatments.



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			

Table 5: Risk Treatment Table

After the application of the CORAS framework on the eVoting system, the following conclusions were reached:

1) 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.

2) 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.). Moreover, the analysis revealed the need for emphasis on certain design choices,

[†] The voting protocol supports multiple votes but only the last vote submitted is considered as valid.

such as the user interface, the clear communication between the voter and EA through precise messages, reasonable time intervals between two subsequent votes etc.

3) The need for extensive checking for faults emerged, e.g. by a body of external auditors who should have at their disposal all the details of the system (even the module's source code). Also, some elements were identified which are crucial for the overall system security, and which should receive particular attention: use of certified servers for time-stamping and cryptographically strong random number generators, cross-validation among different communication modules etc.

4) The need for taking physical and organizational measures, apart from the technical ones, was made apparent, in order to avoid malicious interventions with the proper system operation. The analysis prompted us to take measures such as the use of secure USB tokens or smart cards, for critical data such as the secret shares of the global key.

It should be noted that, before the system is ever used in order to conduct a real, critical election process, the risk analysis process should be performed at a finer and deeper detail taking into account, also, and the surrounding infrastructure.

5. Discussion

The application of CORAS to the eVoting domain has led us to useful conclusions, about the capabilities of handling the security issues of our target system.

With regard to *effectiveness*, CORAS was found to:

- *Increase the probability of locating many significant security threats:* The framework provides the developers of the system with a systematic way of finding all security risks and handling them at an early stage.
- *Facilitate the precise and unambiguous description of the security relevant features and risk analysis results:* The quality of risk analysis depends on precise descriptions of the target system, its context and security relevant features like threats. This, for example, was made apparent with the voting protocol that was analyzed using a time sequence diagram, in which it was made clear how the different steps of the protocol interact with each other at the implementation level.
- *Reduce the probability that risks are overlooked as a result of misconceptions:* The use of easily understandable graphical descriptions as a medium for communication and interaction between stakeholders involved in a risk analysis session, as well as the management of the target system, reduces the probability for misconceptions. This is achieved much in the same way that mathematical modeling provides a precise and unambiguous description of an observed phenomenon or process.

With regards to *performance*, CORAS can:

- *Reduce maintenance costs*: The modeling technology and the support by computerized tools facilitate a fast, more precise and unambiguous documentation of the risk analysis results and the assumptions on which their validity depend. This should increase the possibilities for cost-effective reuse of risk analysis results during maintenance.
- *Reduce system development costs:* CORAS provides a tight integration of risk management in the system development process. Overall, CORAS offers another approach to risk analysis than the traditional ones, where risk analysis often is a separate activity.

With regards to *usability*, CORAS's guidelines for the use of modeling technology make it easier to provide input to risk analysis, to document the results of the analysis, and to present the output to the relevant stakeholders in a consistent and understandable way.

We believe that the observations above point to the fact that the application of a formal risk analysis framework like CORAS, should not be too hard to apply and can lead to a variety of benefits for the target system resulting, eventually, to enhanced *system transparency* and *wider system acceptance* by the public as well as external system auditors alike.

References

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

[2] Andrews, J.D and Moss, T.R., *Reliability and Risk Assessment*, Longman Group UK, 1993.

[3] 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.

[4] Australian Standard: *Risk Management, AS/NZS 4360:1999,* Strathfield: Standards Australia, 1999.

[5] Barber, B., and Davey, J., *The Use of the CCTA Risk Analysis and Management Methodology CRAMM in Health Information Systems,* in MEDINFO 92, edited by Lun K C, Degoulet P, Piemme T E and Rienhoff O., North Holland Publishing Co, 1992, pp.1589 - 1593.

[6] British Standard BS 5760: Reliability of systems, equipment and components, Part 7: "Guide to fault tree analysis", 1991.

[7] Bouti, A., and Ait Kadi, D., *A state-of-the-art review of FMEA/FMECA,* International Journal of Reliability, Quality and Safety Engineering, 1 (4), 1994, pp. 515-543.

[8] Grose, T. J, Doney, G. C, Brodsky, S. A., *Mastering XMI: Java Programming with XMI, XML, and UML*, Wiley, 2002.

[9] Kletz, T., Hazop and Hazan: Identifying and assessing process industry hazards, Taylor & Francis, 4th Edition, 1999.

[10] Krutchten, P.,*The Rational Unified Process, An Introduction*, Reading, Addison-Wesley, 1999.

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

[12] Siu, N., *Risk Assessment for dynamic systems: An overview*, Reliability Engineering and System Safety, Vol. 43, 1994, pp. 43-73.

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

[14] 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.