23rd Bled eConference

eTrust:

Implications for the Individual, Enterprises and Society

June 20 - 23, 2010; Bled, Slovenia

Measuring eTrust in distributed systems
- General Concept and Application to Internet Voting -

Guido Schryen

University of Freiburg, Germany

guido.schryen@is.uni-freiburg.de

Melanie Volkamer

Technische Universität Darmstadt / CASED, Germany

melanie.volkamer@cased.de

Abstract

Emerging digital environments and infrastructures, such as distributed services and computing services, have generated new options of communication, information sharing, and resource utilization in past years. Different distributed trust concepts are applied to increase trust in such systems. However, these concepts yield to rather complex architectures which make it difficult to determine which component or system needs to be trusted. This paper presents a novel trust measurement method for distributed systems which enables the t identification of weak points in the overall system architecture. The measurement method includes the specification of a formal trust language and its representation by means of propositional logic formulas. The applicability of the proposed concepts is demonstrated by conducting a case study on the Internet voting system that was used in the 2007 parliamentary elections in Estonia.

Keywords: distributed trust concepts, measuring etrust, Internet voting

1 Introduction

Distribution concepts are nowadays used in many different contexts in order to increase quality and trustworthiness of services. Examples are grid computing, cloud computing, and web services. Distribution in the context of trustworthiness is applied to overcome single trusted third parties. The idea is to distribute tasks to different parties so that a single party is not able to violate requirements. However, in a situation of maliciously collaborating parties, they are able to violate requirements. According to (Volkamer, Grimm, 2009), three different types of distributed trust concepts can be distinguished: separation of duty, four eyes principles and multiplicity of control function. These

approaches can be applied in different ways depending on the addressed requirements. They are also combined in arbitrary ways in one application or service. This is in particular the case in complex and security critical applications. For instance in Internet voting systems usually the following concepts are applied: the separation of duty concept is implemented for the distribution of voting servers, the four eyes principle for voting server administrators and the election commission, and the multiplicity of control functions for tallying and re-tallying. In the presence of many different components (servers, software, and persons) it is not clear which component needs to be trusted to not maliciously cooperate with other component(s) in order to violate requirements.

Current, IT security evaluation standards like the Common Criteria or IT-Grundschutz (IS0 27001) do not support distributed trust concepts properly. First, they focus on outsider threats and neglect insider threats while distributed trust concepts try to overcome the latter. Second, they do not consider that distributed trust concepts are very flexible (e.g., in a MIX net you can easily add one MIX component to increase the trustworthiness regarding anonymity), but only address systems that are static in their architecture and implementation (e.g., one MIX node). However, for many applications, such as Internet voting, it is important to determine the level of trust in terms of whom to trust not to maliciously collaborate with others[1].

In this paper we introduce a formal trust language to describe distributed systems with regard to the protection against insider threats (parties who might violate requirements by cooperating maliciously). The language is based on the idea of *k* resilience terms introduced in (Volkamer and Grimm, 2009). To exploit the expressiveness of the proposed trust language we further propose to derive propositional trust terms, which allow for the identification of weak points, the suggestion of improvements, and the comparison of different systems.

As voting is one of the most critical applications, we apply our concepts in a case study on the Estonian remote electronic voting system in the setting of the 2007 nationwide parliamentary election. We show how the concepts can be applied in a real-world environment to analyze systems regarding their trust properties and to identify weak points.

The remainder of this paper is structured as follows: Section 2 presents related work. In Section 3, we provide our research framework. Section 4 proposes the formal trust language, and Section 5 shows the mapping of trust language terms on propositional logic terms. Section 6 applies the theoretical concepts on the Estonian Internet voting system, before Section 7 concludes the paper.

2 Related work

There is a substantial body of literature on concepts, models, evaluation, and management of trust in digital environments (see (Ries, 2009) for a detailed overview). Analyzing this body, (Krukow and Nielsen, 2007) identifies two lines of research: The first strand is based on a technical understanding coined by (Blaze et al. 1996) and includes the “access control list" approach and the “credential-based" approach. The second strand is “experience-based" and assumes that an entity's trust in another is based on the others’ past behaviour. Reputation-based approaches and the techniques proposed in this paper are examples of this strand (Krukow and Nielsen, 2007).

[1] Insider threats are already addressed in polling place elections, where poll workers observe each other in order to prevent that election requirements are violated by an individual.

However, despite the comprehensive literature on trust, its measurement and metrics have been addressed very rarely only. (Kohlas et al., 2008) is a valuable example and presents a trust evaluation model that is based on logic and probability theory.

Trust in electronic voting (systems) can be increased through various measures. (Volkamer, 2009) suggests to evaluate and to certify electronic voting software according to the Common Criteria (2006). (Schmidt et al., 2009) recommend extending this evaluation by an IT-Grundschutz certification (BSI, 2005). In both cases, trust can be measured according to the covered requirements, the addressed intruder models, and the chosen evaluation levels. However, both evaluations primarily address attacks from the outside while this paper addresses insider threats.

3 Research framework

An overview of our research approach is shown in Figure 1. We define distributed systems inductively, i.e. a distributed system is either an atomic system or is composed of other (sub)systems. We define a system as atomic if it contains only (atomic) components that are not being split any further. These components can be persons, computers, or even organizational units.

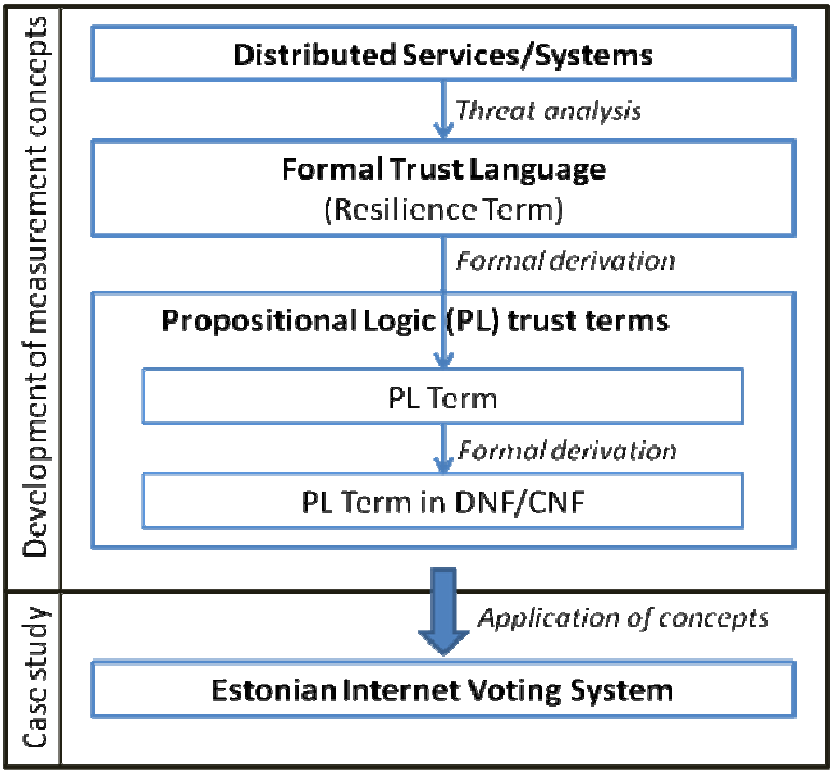

Figure 1: Research Framework

The trust language we use in this paper draws on (Verheul and van Tilborg, 1997; Hofmeister et al., 2000) , who use secret shares (Blakley and Kabatiansky, 2005) and the concept that *k* out of *n* entities are required for revealing the secret. We adapt this concept in the context of trust, and say: "*k* out of *N* entities must be trusted to not cooperate maliciously". In contrast to the aforementioned papers, we explicitly use the set of entities *N* in order to explicitly account for the heterogeneity of entities in *N* (compare to (Volkamer, 2009)). Based on this understanding, we propose a formal trust language. We refer to elements of this language as "resilience terms", which formally describe trust properties of a system (with regard to its robustness against insider threats).

According to secret sharing, where we need to trust *k* out of *n* entities regarding secrecy and *(n-k)* out of *n* entities regarding availability, different security requirements on a system also lead to different resilience terms. Thereby, this language also allows us to show that some distributed trust concepts which were introduced to increase the trust regarding one particular requirement contemporaneously decrease the trust regarding other requirements. For example, adding one MIX in a simple decryption MIX net increases the trustworthiness regarding secrecy while it decreases it regarding availability.

While resilience terms are a useful representation for expressing robustness against insider threats, they are less appropriate for analyzing and comparing systems. We show (1) how resilience terms can be mapped on equivalent propositional logic terms and (2) that the transformation of arbitrary propositional logic terms in conjunctive normal form (CNF) and in disjunctive normal form (DNF) is a useful way to overcome these limitations.

Finally, we apply our concepts in a case study.

4 Formal trust language

The definition of the formal trust language (resilience terms) in terms of syntax and semantics follows the inductive definition of systems. Both are provided by definitions 1-6. In order to keep definitions short, we introduce the abbreviation wrts. *r* (with regard to security requirement *r*).

Let *S* be an atomic system with its set of atomic component $A = \{A_i\}_{i=1}^{n}$.

Definition 1: *A system S is (k out of N) – resilient,* $k \in \{1,...,|N|\}, k < |N|, N \subseteq A$, *wrts. r*

$$:\Leftrightarrow \begin{cases} \textit{At least k components out of N need to be trusted to not cooperate} \\ \textit{maliciously (with other components of N) wrts. r in order to make S} \\ \textit{meet r.} \end{cases}$$

Definition 2: *A system S is (1 out of N)-resilient,* $|N| = 1, N \subseteq A$, *wrts. r*

$$:\Leftrightarrow \begin{cases} \textit{One need to trust the one component in N wrts. r in order to make S} \\ \textit{meet r.} \end{cases}$$

Remark 1. Resilience terms like (*1* out of *{A_i}*) contradict the idea of distributed trust concepts. However, it needs to be included because distributed trust concepts are not always perfectly applied. Moreover, distributed trust concepts implemented to improve the trustworthiness of a system regarding a particular requirement usually weaken other requirements if no additional mechanisms are implemented. In this case, such resilience terms can occur.

In order to get more flexible representations of requirements on atomic systems, we define the following resilience terms:

Definition 3: *A system S is a)* $((k_1 \wedge ... \wedge k_m)$ *out of* $(N_1,..., N_m))$ *-resilient wrts. r, b)* $((k_1 \vee ... \vee k_m)$ *out of* $(N_1,..., N_m)$ *- resilient wrts. r, where* $\forall i : k_i \in \{1,...,|N_i|\}$, $(k_i < |N_i| \text{ or } k_i = 1)$, $N_i \subseteq A$

$$:\Leftrightarrow \begin{cases} \textit{For a) each b) any i} \in \{1,...m\}, \textit{ at least } k_i \textit{components out of } N_i \textit{need} \\ \textit{need to be trusted to not cooperate maliciously (with other components} \\ \textit{of } N_i) \textit{ wrts. r so that S meets requirement r.} \end{cases}$$

Remark 2. Resilience terms like (*2* out of $\{A, B\}$) are not meaningful, because one either needs to trust that at least one of these two components is trustworthy and does not maliciously cooperate with the other one (*1* out of $\{A, B\}$), or one need to trust both components because each component itself can violate the addressed requirement ((*1* $\wedge$ *1*) out of ($\{A\},\{B\}$)).

Small example: An onion MIX net (comp. to (Chaum, 1981)) with n nodes $N = \{N_1,..., N_n\}$is (*1* out of *N*) resilient with respect to anonymity. The resilience term regarding availability is ((*1* $\wedge ... \wedge$ *1*) out of ($N_1,..., N_n$)).

With regard to non-atomic systems, we define resilience terms similarly: Let $\{S_i\}_{i=1}^{n}$ be (sub)systems of a system *S*, and let system S_i be l_i -resilient for all $i \in \{1,...,n\}$.

Definition 4: *A system S is (k out of* $\{l_{i_1},...,l_{i_m}\}$*)-resilient,* $k \in \{1,...,m\}$, $k < m$, $\{i_1,..., i_m\} \subseteq \{1,..., n\}$, *wrts. r*

$$:\Leftrightarrow \begin{cases} \text{At least k systems out of } \{S_{i_1},...S_{i_m}\} \text{ need to be trusted to not co-} \\ \text{operate maliciously (with other components of } \{S_{i_1},...S_{i_m}\}) \text{ wrts. r} \\ \text{in order to make S meet r.} \end{cases}$$

Definition 5: *A system S is (1 out of* $\{l_j\}$*)-resilient,* $\{j\} \subseteq \{1,..., n\}$, *wrts. r*

$$:\Leftrightarrow \begin{cases} \text{One need to trust the system } S_j \text{ (with resilience term } l_j \text{) wrts. r in} \\ \text{order to make S meet r.} \end{cases}$$

Definition 6: *A system S is a)* $((k_1 \wedge ... \wedge k_m)$ *out of* $(N_1,..., N_m))$*-resilient wrts r, b)* $((k_1 \vee ... \vee k_m)$ *out of* $(N_1,..., N_m))$*-resilient, where* $\forall i : k_i \in \{1,...,|N_i|\}$, $N_i \subseteq \{l_1,..., l_n\}$ $(k_i < |N_i|$ *or* $k_i = N_i = 1)$

$$:\Leftrightarrow \begin{cases} \text{For a) each, b) any } i \in \{1,...m\} \text{ at least } k_i \text{ systems out of the set} \\ \text{of systems for which Ni contains resilience terms need to be trusted} \\ \text{to not cooperate maliciously (with other systems of the set) wrts. r} \\ \text{in order to make S meet r.} \end{cases}$$

5 Derivation of propositional logic trust terms

Resilience terms can become complex, even for small systems. In order to yield representations that are even for large systems comfortable to interpret for persons and appropriate for the computation of the uncertainty with which a system does not fulfil a specific requirement r, we transform resilience terms into propositional logic formulas: Let system *S* consists of basic components $\{A_1,..., A_n\}$, and let $\{X_{A_1},...,X_{A_n}\}$ be literals with $X_{A_i} = true\ \forall i$, *iff* A_i is trustworthy. Then, the resilience term *l* of a system S can be mapped on a propositional logic formula *f (l)* such that *S* is trustworthy *iff f(l)* is true.

This can be proven along the inductive definition of resilience terms. However, the formal proof is skipped due to space reasons. The principal idea of the proof is that we reformulate the expression "*k* out of a set *L*" by explicitly considering all combinations of elements of *L*, where *L* can be either a set of basic components or of subsystems. The provision of such a mapping *f* (of resilience terms on propositional logic terms) proves the above theorem.

Small example: Assuming that the *k* resilience term of a particular system is $((1 \wedge 2)$ out of $(\{A, B\},\{C, D, E\})$). Then the corresponding logical formula is (A $\vee$ B) $\wedge$ ((C $\wedge$ D) $\vee$ (C $\wedge$ E) $\vee$ (D $\wedge$ E)).

Particularly useful is the subsequent transformation of the formulas into semantically equivalent formulas in normal form, such as the disjunctive normal form (DNF) or the conjunctive normal form (CNF). These normal forms feature different strengths: the CNF allows determining "weak points", such as single points of failure. For instance, assuming the CNF has the form $A \wedge (B \vee C)$. Then, component A is the single point of failure. This component needs to be trusted because it can - without any malicious cooperation – violate the corresponding requirement. Obviously, the system should be improved here by applying a corresponding distributed trust concept. The DNF is useful for identifying "strong points", such as components or subsystems where their trustworthiness results in the trustworthiness of the overall system, regardless of the trustworthiness of other components and subsystems. For instance, let us assume that the CNF has the form $A \vee (B \wedge D) \vee C \vee E$. Such long chains of "$\vee$" connections represent very trustworthy systems as it is enough to either trust A or C or E or both B and D. Thus, both normal forms should be applied complementarily.

Remark 3: If the trust term in CNF of one subsystem of system S contains the component A and the CNF trust term of another subsystem contains $A \vee B$, then the CNF trust term of S only contains A. That is, if in one subsystem component A is involved and one need to trust this one component while in the other subsystem it needs to be trusted that either component A or component B meets the requirement, then for the composition of these two subsystems B is not relevant any more as $A \wedge (A \vee B) = A$.

6 Case study

In this chapter we apply the proposed theoretical concepts by determining k-resilience terms, propositional logic terms, and the disjunctive/conjunctive normal forms in a real-world Internet voting setting. We analyze the Estonian Internet voting system, which was used in 2007 as first nation-wide parliamentary election. Due to time and space restrictions, we apply our approach only on two selected requirements on remote electronic voting systems (for a comprehensive list of requirements see (CoE, 2004)):

- *Secrecy of the vote*: It is not possible to establish a link between voter and his/her plaintext vote (without the support of a voter).
- *Integrity of the election result:* It is not possible to modify the election result undetected by (a) removing, (b) adding or (c) altering votes during vote casting or in the electronic ballot box[2]. Furthermore, it is not possible to modify the result undetected by (d) manipulating the tallying algorithm.

The analysis is done in the following way: We first describe the system and sketch its architecture with regard to the trust concepts. Based on this, the *k*-resilience terms and the corresponding propositional logic terms are determined. Finally, we present our findings.

6.1 System Description

The Estonian Internet voting system (compare to (OSCE, 2007; Maaten, 2004; Madise and Martens, 2006)) essentially implements the digital analogon of postal voting: That is, the voter encrypts his vote (inner envelope) and then signs this encrypted vote using his/her digital identity card (outer envelope). This encrypted and signed vote is sent to

[2] Note, we do not consider attacks by untrusted voter PCs.

the Voting Server[3]. All signed encrypted votes are stored in an electronic ballot box on one voting server. After the vote casting period, the voting software checks (in an offline mode) whether only votes from eligible voters are stored (including only one vote per voter). Afterwards, this software removes the signatures and shuffles the encrypted votes. This new list of (anonymous) encrypted votes is burned on a CD by the administrators. Then the CD box is sealed and taken to the (offline) counting component. This is done by one of the election officials. The seal is tamper resistant and its integrity is checked during the public tallying.

The counting component is connected to a Hardware Security Module (HSM), which securely stores the decryption key. After having enabled this module by four different physical keys, the counting component sends the encrypted votes to the HSM (vote by vote) and receives the decrypted votes (vote by vote). Based on this output, the software computes the election result. The result and the decrypted votes (order of output from the HSM) are public information. While the physical keys were held by five election officials (O_1, O_2, O_3, O_4, O_5) , all other tasks, including server and system installation were done by two administrators (A_1, A_2) who both had to enter their passwords for any action after the system setup. The whole system with the exception of the HSM was developed by a single company (software developer *SD* and HSM developer *HD*). Figure 2 shows a simplified version of the system architecture.

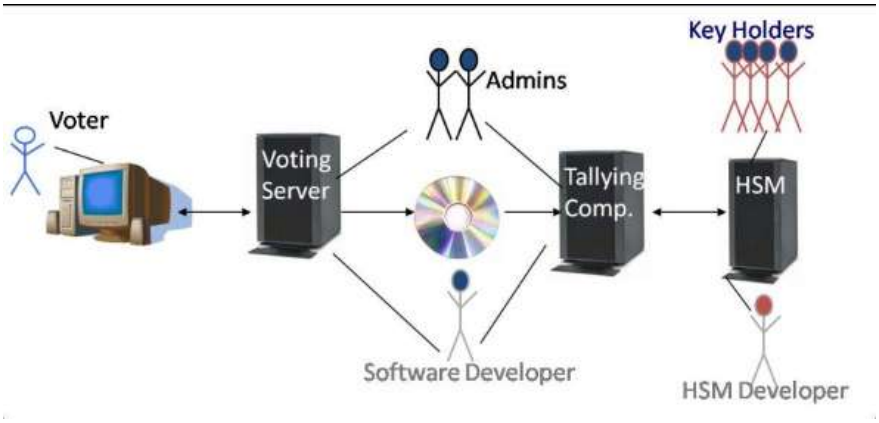

Figure 2: Estonian Internet voting system setting

6.2 System Analysis

In this section we analysis the Estonian Internet voting system with our framework for the two selected requirements: secrecy of the vote and integrity of the election result:

Secrecy of the vote

The secrecy of the vote can only be violated by the subsystem "Voting Server". Later on, at the tallying component and the Hardware Security Module, the votes are being anonymized if the Voting Server and in particular its shuffling are trustworthy. Thus, the Voting Server needs to be trusted regarding the secrecy of the vote in order to meet this requirement with the whole voting system. Even if the Tallying Component and the HSM would maliciously cooperate they could not break the election result. The *k*-resilience term is: 1 out of $\{VS\}$. *VS* is here the short cut for the *k*-resilience term of the Voting Server itself.

[3] Vote updating was enabled for the Estonian parliamentary elections. However, as this functionality does not influence the *k*-resilience value, it is not further considered.

At the side of the Voting Server, it needs to be trusted that one of the administrators A_1, A_2 is trustworthy and also the software developer *SD*. They could implement or manipulate the voting software on the Voting Server: The easiest way is to disable the shuffling before burning the CD (because of the publication of all decrypted votes in the same order as they are in an encrypted form on the CD). According to definition 3 this is *(1∧1 out of {A1, A2}, {SD})*. According to definition 5, the k-resilience term of the whole electronic voting system is 1 out of *(1∧1 out of {A1, A2}, {SD})*. Compare to Figure 3 for the development of the *k*-resilience value.

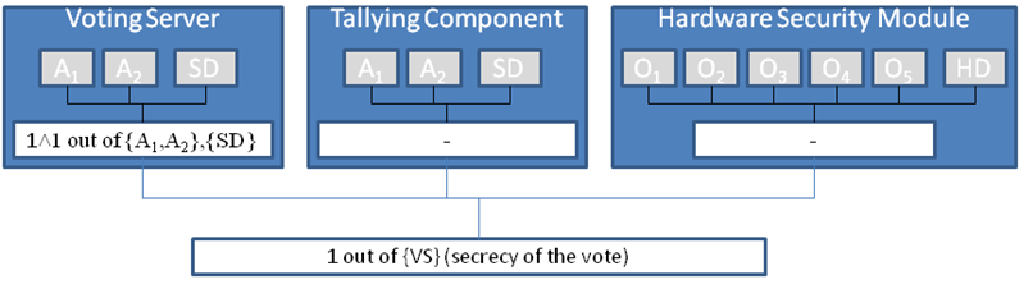

Figure 3: k-resilience value of the Estonian voting system with regard to the requirement "secrecy of the vote"

This *k* resilience term corresponds to the following propositional logic term: $(A_1 \vee A_2) \wedge SD$. This is already in conjunctive normal formal. The corresponding DNF is $(A_1 \wedge SD) \vee (A_2 \wedge SD)$.

Integrity of the election result

All three components could undetectably modify the election result and thus violate its integrity: The Voting Server could provide a malicious voting applet, which does not encrypt and sign (with the voter's secret key) the vote as intended by the voter but modifies it before the encryption (for instance, if the Voting Server is in favour of candidate A, but the voter chooses candidate B, then the voting applet is manipulated in a way that it always encrypt candidate A). The Tallying Component could output a predefined election result, which is independent of the output from the Hardware Security Module and which only takes the total number of cast votes into account. Similarly, the HSM could output decrypted votes which are independent from the input encrypted votes. This overall trust concept corresponds to *(1 ∧ 1 ∧ 1 out of {VS}, {TC}, {HSM})* (according to definition 3). *VS* is the shortcut for the k-resilience value of the Voting Server, TC of the Tallying Component, and HSM of the Hardware Security Module component.

At the Voting Server, it needs to be trusted that one of the administrators A_1, A_2 is trustworthy and the software developer *SD*. If they are malicious, they could implement or manipulate the provided voting applet on the Voting Server: According to definition 3, this is *1∧1 out of {A1, A2}, {SD}*. The same holds for the Tallying Component because the same persons are involved (only the attack is different). At the Hardware Security Module, it needs to be trusted that the developer HD is trustworthy. According to definition 2, this is 1 out of {HD}.

According to definition 6, the *k*-resilience term of the whole electronic voting system is *(1 ∧ 1 ∧ 1 out of (1∧1 out of {A1, A2}, {SD}), (1∧1 out of {A1, A2}, {SD}), (1 out of {HD}))*. Compare to Figure 4 for the development of the *k*-resilience value.

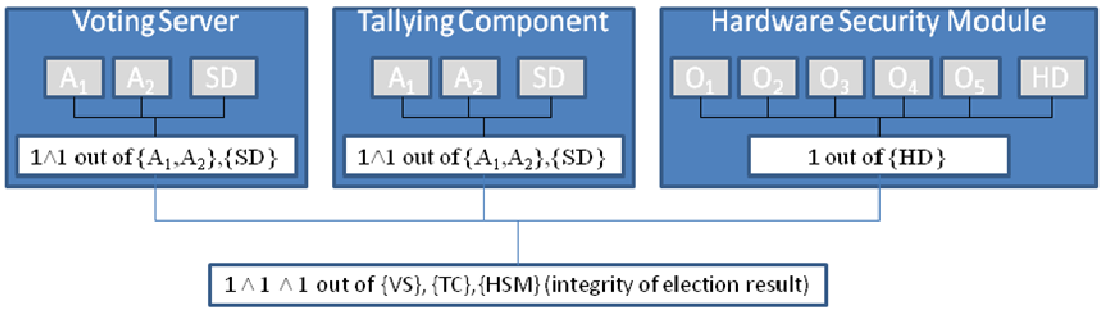

Figure 4: Estonian voting system's k-resilience value for the requirement "integrity of the election result"

This *k* resilience term corresponds to the following propositional logic term*:* $(A_1 \vee A_2) \wedge SD \wedge HD$. This is already in conjunctive normal formal. The corresponding DNF is $(A_1 \wedge SD \wedge HD) \vee (A_2 \wedge SD \wedge HD)$.

6.3 Findings and possible improvements

The conjunctive normal forms show that there is one single point of failure (the developer of the electronic voting software) regarding the secrecy of the vote and, even worse, there are two single points of failure regarding integrity (the developer of the electronic voting software and the developer of the Hardware Security Module). Furthermore, the conjunctive normal forms make clear that it is important that both administrators control each other and have different interests. If they would maliciously collaborate they could also violate both analysis requirements.

In order to improve the situation there are several technical possibilities to implement universal and individual verifiability, which enables the voter to audit whether his/her vote is properly cast, stored and tallied as well as to enable the public to verify that only authorized votes are tallied and that these are tallied correctly. It is also possible to increase the trust in these single points. Regarding the Hardware Security Module, one could buy one module from each company that sells these modules and then randomly choose one for the election. Regarding the software developer of the voting software, a couple of different measures might improve the situation: One could publish the source code. Further, the software company should apply pair programming and carefully log who has access to the source code and could modify it.

Any of these measurements would improve the situation. However, it is necessary to identify single points of failure in order to come up with corresponding measurements.

Indirectly, the logic formulas lead to a second important finding: It is often claimed that key holders ensure the secrecy of the vote. However, as O_i does neither appear in the *k*-resilience term nor in the logic term, the key holders do neither protect nor are they able to violate one of the analyzed requirements. That is, even if they all maliciously cooperate, they are not able to break the secrecy of the vote. Moreover, it is also not required to have access to the HSM decryption key in order to violate the secrecy of the vote.

7 Discussion and Conclusion

This paper presents a formal approach towards the measurement of trust in distributed systems. While we focus in this paper on systems that implement distributed trust concepts, the trust language and the propositional logic terms can be applied on any distributed system in order to measure the trustworthiness regarding particular requirements. These requirements can envelope – besides security requirements – other requirements, such as efficiency and quality of service.

We apply our etrust measurement to the Estonian Internet voting system in the setting of the 2007 nationwide parliamentary election. Our measurement enables us to identify single points of failures (in particular by using the conjunctive normal form). Based on these findings, we also provide concrete procedures to overcome these single points of failures and to reduce the risk that these single points become malicious. Furthermore, we show that one of the implemented distributed trust concepts does not have an effect on the *k*-resilience terns with regard to the two analyzed requirements. As the election officials are involved in this step, they might be surprised that neither the secrecy of the vote nor the integrity of the election result depends on the keys they share. Obviously, such results will not be achieved by standard IT security evaluations. However, as our approach mainly addresses insider threats, we recommend applying both existing evaluation standards and our approach. Furthermore, persons in charge for IT security critical systems should consider additional measures to increase the trust in their system, in particular in the context of electronic voting systems. Such measures would include the publication of source code and the implementation of universal and individual verifiability.

A valuable extension of our work would be the application of probability theory in order to quantitatively determine the overall certainty of a system with regard to a particular security requirement. Assigning each literal (component) in the propositional logic term a probability (of failure) value, we can aggregate probabilities according to the way literals are arranged in the propositional logic term. This approach would support the design of trust metrics.

References

Adida, B., (2008). Web-based open audit voting. Source Proceedings of the 17th conference on Security symposium table of contents (335 – 348). USENIX Association Berkeley, CA, USA.

Adida, B., de Marneffe, O., Pereira, O., Quisquater, J. (2009). Electing a University President Using Open-Audit Voting: Analysis of Real-World Use of Helios, In D. Jefferson, J.L. Hall, T. Moran, editor(s), Electronic Voting Technology Workshop/Workshop on Trustworthy Elections, Usenix.

Blakley, R., Kabatiansky, G. (2005). Secret Sharing Schemes. Encyclopedia of Cryptography and Security (544-545). Springer.

Blaze, M., Feigenbaum, J., Lacy, J. (1996). Decentralized trust management.In Proceedings from the 17th Symposium on Security and Privacy (164-173) IEEE Computer Society Press.

Chaum, D. (1981). Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. Communication of the ACM 24(2), (84–90).

Common Criteria for Information Technology Security Evaluation (CC) is an international standard (ISO/IEC 15408) for computer security evaluation and certification, 2006 http://www.commoncriteriaportal.org/.

Council of Europe (CoE) Legal, operational and technical standards for e-voting. Recommendation rec(2004)11 adopted by the committee of ministers of the Council of Europe and explanatory memorandum. Strasbourg.

German Federal Office for Information Security BSI (2005). IT–Grundschutz Catalogues, http://www.bsi.de/english/gshb/download/it-grundschutz-kataloge_2005_pdf_en.zip.

Hofmeister, T., Krause M., Simon H.U (2000). Optimal k out of n secret sharing schemes invisual cryptography. Theoretical Computer Science (240-471).

Kohlas, R., Jonczy, J., Haenni, R. (2008). A trust evaluation method based on logic and probability theory. In P. Herrmann Y. Karabulut, J. Mitchell and C. D. Jensen, editors, 2nd Joint iTrust and PST Conferences on Privacy, Trust Management and Security, volume II of Trust Management, (17-32), Trondheim, Norway.

Krukow, K., Nielsen, M. (2007). Trust structures: Denotational and operational semantics. International Journal of Information Security, 6(2-3) (153-181).

Maaten, E. (2004). Towards Remote E-Voting: Estonian Case. In: Electronic Voting in Europe, (83–100).

Madise, U., Martens, T. (2006). E-Voting in Estonia 2005. The First Practice of Country-wide Binding Internet Voting in the World. In: Krimmer, R. (ed.) Electronic Voting 2006, 2nd International Workshop, LNI, vol. 86, (15–26).

Office for Democratic Institutions and Human Rights (2007). Republic of Estonia – Parliamentary Elections – 4 March 2007 – OSCE/ODIHR Election Assessment Mission Report. Organization for Security and Co-operation in Europe.

Ries, S. (2009). Trust in Ubiquitous Computing. PhD Thesis at Technische Universität Darmstadt.

Schmidt, A., Heinson, D., Langer, L., Opitz-Talidou, Z., Richter, P., Volkamer, M., Buchmann, J. (2009). Developing a Legal Framework for Remote Electronic Voting**.** In Proceedings of VOTE-ID 2009, volume 5767 of LNCS, pages 92-105, Berlin/Heidelberg, Springer-Verlag.

Verheul, E. R., van Tilborg H. C. A. (1997). Constructions and properties of k out of n visual secret sharing schemes. Designs, Codes and Cryptography, 11(2):(179-196).

Volkamer, M. (2009). Evaluation of Electronic Voting? Requirements and Evaluation Procedures to Support Responsible Election Authorities, volume 30 of LNBIP. Springer, Berlin, Heidelberg.

Volkamer, M., Grimm, R. (not published yet). Determine the Resilience of Evaluated Internet Voting Systems**.** In Postproceedings of Re-Vote09 (http://www-public.it-sudparis.eu/~gibson/RE-Vote09/).